KU LEUVEN

ARENBERG DOCTORAL SCHOOL
Faculty of Science

PCA-based Methods for Monitoring High-Dimensional, Time-Dependent Processes

Eric Schmitt

Supervisors:
Prof. dr. M. Hubert
Dr. ir. B. De Ketelaere

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor in Science: Statistics

September, 2016

PCA-based Methods for Monitoring High-Dimensional, Time-Dependent Processes

Eric SCHMITT

Examination committee:
Prof. dr. A. Carbonez, chair
Prof. dr. M. Hubert, supervisor
Dr. ir. B. De Ketelaere, supervisor
Prof. dr. Alberto Ferrer (Technical University of Valencia)
Prof. dr. Martina Vandebroek
Prof. dr. Wouter Saeys
Prof. dr. Tim Verdonck

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor in Science: Statistics

September, 2016

Uitgegeven in eigen beheer, Eric Schmitt, Celestijnenlaan 200B, B-3001 Leuven (Belgium)

Preface

This dissertation sets out to document the literature on PCA-based statistical process monitoring for time-dependent data and address gaps in understanding, specifying and applying these methods. The doctorate begins with a literature review, and ends with a toolbox delivering easy implementation of the methods, with features that cover many of the gaps identified in the intervening chapters. The work done in this dissertation was rarely done alone. On all of the papers, I did the majority or totality of the writing and much of the statistical analyses, but in the papers with Tiago Rato and Tom Reynkens, we shared these duties, improving the overall quality of the papers greatly. Prof. Hubert and Dr. De Ketelaere provided excellent supervision of these projects.

As alluded to above, a project of this scope would not have been possible without the support and contributions of many talented people. My promoters, Prof. Mia Hubert and Bart De Ketelaere have acted as excellent mentors throughout this process and provided wonderful guidance. I would also like to extend my thanks to Tiago Rato, who I had the great fortune of collaborating with on many of the papers that compose this dissertation, and with whom it was a pleasure to work. Tom Reynkens was also a fantastic collaborator, and work on the ROSPCA paper was both productive and enjoyable because of his substantial contributions.

In addition to the papers written for the dissertation, my PhD years involved the writing and publication of a number of independent articles. I would like to thank Vivien Marasigan for the opportunity to amuse myself with Bayesian meta-analysis on the topic of skin cancer as well as her friendship. Machteld Vandecandelaere also deserves my gratitude for the opportunity to work on two very interesting education papers together, as well as my praise for her excellent work. Resulting from collaboration with Waichun Leong, I had the unexpected pleasure to present on the philosophy of aesthetics at a conference. I would like to thank Patrick and Drew Atwater for the privilege of writing a resource management paper together in connection with the California draught, and

Patrick for the challenges and adventures that the California Data Collaborative (a multi-district data utility), which grew out of our paper regularly, continues to visit upon me. Among my extra-curricular co-authors, Kaveh Vakili stands out as the most prolific, and has been the most influential on my work.

In addition to academic support, I also sincerely thank my wife, Lieselot for her years of love and support and the jokes.

I acknowledge the following funding sources: Flanders Agency for Innovation by Science and Technology (IWT) through the SBO POM project, and the Industrial Research Fund of the KU Leuven, the Internal Fund of the KU Leuven and the SCISSOR ICT project no. 644425, funded by the European Commission through the Information & communication technology H2020 Framework Program.

Abstract

The focus of statistical process monitoring (SPM) is the development of statistical models of processes that allow operators to identify when a process is behaving atypically and pinpoint the sources of this behavior. When the process is high-dimensional (e.g. when many sensors take measurements on an industrial process), the modelling process can be complicated by both the high number of variables, and collinearities that typically arise between them. Conventional process monitoring approaches typically grow quickly in complexity with the number of variables being monitored. Furthermore, the calculation of model parameters and control limits usually requires the inversion of a covariance matrix, which collinearities can render singular. These two drawbacks have motivated the introduction of SPM methods based on latent variable methods such as Principal Component Analysis (PCA) and Partial Least Squares (PLS). Both approaches first transform the data to a low-dimensional subspace before conducting complex analysis. However, both of these approaches were originally developed to analyze independently and identically distributed data. In practice, processes frequently exhibit autocorrelation and non-stationarity. The dissertation will focus on applications of PCA in this context.

This dissertation reviews the existing literature on latent variable-based process monitoring and a number of gaps are identified. First amongst these is the lack of a comprehensive examination of their behavior on the types of processes they were designed to monitor. Two chapters in the dissertation treat this issue. These chapters also introduce another gap in the existing literature, which is the absence of guidance for how to parametrize these methods. This topic is treated in its own chapter. Finally, one of the important assumptions of these methods is that the data the monitoring model is fitted on, is representative of normal operating conditions, but this assumption is often violated due to faults in the training and calibration data. Addressing this issue, as well as other issues of interest to outlier detection, a robust, sparse PCA method is introduced in its own chapter.

A major challenge for practitioners and academics working with latent variable methods in process monitoring is that there is no software tool that combines the most important methods in a user friendly and extensible form. To address this issue, a Matlab toolbox for multivariate process monitoring, with emphasis on high-dimensional data is introduced in appendix.

Beknopte samenvatting

De focus van statistische procesopvolging (SPM) is de ontwikkeling van statistische modellen van processen die gebruikers toelaten vast te stellen wanneer een proces zich atypisch gedraagt en wat de redenen van dit gedrag zijn. Wanneer het proces hoogdimensionaal is (bijvoorbeeld wanneer een groot aantal sensoren gebruikt worden), kan het modelleringsproces worden bemoeilijkt door zowel het grote aantal variabelen als door de collineariteiten die tussen de variabelen ontstaan. Conventionele methoden van procesopvolging worden doorgaans snel complexer naarmate het aantal gecontroleerde variabelen toeneemt. Bovendien vereist de berekening van modelparameters en controlelimieten vrijwel steeds het inverteren van een covariantiematrix die omwille van collineariteiten singulier kan worden. Deze twee nadelen motiveren de invoering van SPM-methoden op basis van latente variabelen zoals Principale Component Analyse (PCA) en Partiële Kleinste Kwadraten (PLS). Beide methoden transformeren de gegevens eerst naar een laagdimensionale deelruimte vooraleer de complexe analyse wordt uitgevoerd. Beide methoden werden echter oorspronkelijk ontwikkeld om onafhankelijke en gelijk verdeelde data te analyseren. In de praktijk vertonen processen echter vaak autocorrelatie en niet-stationariteit. Dit proefschrift richt zich op toepassingen van PCA in deze context.

Dit proefschrift bespreekt de bestaande literatuur over procesopvolging gebaseerd op latente variabelen, en identificeert een aantal lacunes. De eerste lacune is het ontbreken van uitgebreid onderzoek van hun gedrag op de soorten processen waarvoor ze ontworpen zijn. Twee hoofdstukken in het proefschrift behandelen deze materie aan de hand van uitgebreide simulaties. Deze hoofdstukken introduceren ook nog een andere hiaat in de bestaande literatuur, namelijk het ontbreken van een leidraad om deze methoden te parametriseren. Dit onderwerp wordt behandeld in een apart hoofdstuk. Ten slotte is een van de belangrijke uitgangspunten van deze methoden dat de gegevens waarop het controlemodel wordt toegepast, representatief zijn voor

het proces onder normale omstandigheden, maar deze aanname wordt vaak geschonden door storingen en uitschieters in de kalibratiegegevens. Om hieraan tegemoet te komen, wordt in een apart hoofdstuk een robuuste, spaarse PCA-methode geïntroduceerd.

Voor mensen uit de praktijk en academici die met latente variabele methoden in procesopvolging werken, is het ontbreken van software die de belangrijkste methoden in een gebruiksvriendelijke en uitbreidbare vorm combineert, een grote uitdaging. Om dit probleem aan te pakken, wordt in de bijlage een Matlab-toolbox voor statistische procesopvolging, met de nadruk op hoogdimensionale data, geïntroduceerd.

Contents

Chapter 1

Introduction

The focus of statistical process monitoring (SPM) is the development of statistical models of processes that allow operators to identify when a process is behaving atypically and the sources of this behavior. When the process is high-dimensional (e.g. when many sensors take measurements on an industrial process), the modelling process can be complicated by both the high number of variables, and collinearities that typically arise between them. Conventional process monitoring approaches typically grow quickly in complexity with the number of variables being monitored. Furthermore, the calculation of model parameters and control limits usually requires the inversion of a covariance matrix, which collinearities can render singular. These two drawbacks have motivated the introduction of SPM methods based on latent variable methods such as Principal Component Analysis (PCA) and Partial Least Squares (PLS). These both approaches first transform the data to a low-dimensional subspace before conducting complex analysis. However, both of these approaches were originally developed to analyze i.i.d. data. In practice, processes frequently exhibit autocorrelation and non-stationarity. The dissertation will focus applications of PCA in this context.

Successful process monitoring relies on models that accurately represent the characteristics of the process of interest. Chapter 2 introduces the high-dimensional, time-dependent SPM context and the specific challenges it poses to achieving this goal. This chapter also provides a general introduction to PCA and variants of it that permit violations of basic assumptions about the observations: in particular autocorrelation and non-stationarity. Each of these variants overcome specific challenges posed by time-dependent processes by generalizing the basic PCA model, but in order to do so they require a more

complex parametrization scheme. Issues that arise as a result are highlighted and investigated more closely in Chapters 3 and 4. In addition to introducing the PCA variants that will be the focus of later chapters, this chapter also discusses extensions to them that can be applicable in special circumstances and analytical tools that support them. In addition to motivating the use of PCA-based methods in a rhetorical fashion by highlighting features of high-dimensional, time-dependent processes that an SPM method must account for, the performance of these methods is explored on a real data example.

Chapter 3 investigates the claims of Chapter 2 more deeply using a comprehensive simulation study of the fault detection capabilities of the methods. Classical PCA and variants designed for auto-correlated and non-stationary data are applied to a range of process types where they are expected to excel or show problems based on claims made in Chapter 2. However, the simulated processes are all parametrized with extreme values to give them exaggerated characteristics, such as very high auto-correlation, or difficult combinations of non-stationarity and autocorrelation. This is done to give a sense of *worst case* performance of the methods since increasing the time-dependence of the process increases the complexity of fitting an accurate model.

A more nuanced approach is taken in Chapter 4, where the parameters of the simulated processes are allowed to take values ranging from "easy" to "difficult." Rather than focusing on fault detection, modeling performance is assessed, with the philosophy that good fault detection begins with good process modeling. The results give a deeper understanding of the reliability of the methods under a spectrum operating conditions. Results of the simulations confirm some expectations expressed in Chapter 2 (and the literature), and reveal the need for a more nuanced understanding of others. In particular, the simulations show that methods which theory suggests should perform best on a given process can encounter difficulties detecting certain types of faults, or do not perform noticeably better than simpler methods they were proposed to replace. In addition to providing a survey of process types and suitable methods for practitioners to refer to when exploring solutions for their problems, the simulation study also details ways that methods can be evaluated on other process types in a principled form. Finally, in the process of building the simulation study, it became apparent that that there was a gap in the literature on how to effectively parametrize PCA and its variants; in particular adaptive methods for non-stationary processes.

The problem of selecting suitable parameters for adaptive PCA models is examined in greater detail in Chapter 5. Two parameters are of particular interest: the forgetting factor determining how quickly the model adjusts to non-stationarity in the process, and parameters for the control limits that give the desired false alarm rate when the model does not completely account for the

structure of the process, which becomes increasingly difficult to accomplish as the process grows more complex. Surprisingly, though adaptive methods are not new to the literature, and the forgetting factor is their most defining parameter, the literature does not provide clear guidance on its selection. Even with a well chosen value for the forgetting parameter, an adaptive PCA model may not produce an accurate enough model to generate i.i.d. monitoring statistics. Residual dynamics in the monitoring statistics require control limits that are capable of accounting for them. A straightforward approach for adjusting the control limits is demonstrated to give monitoring performance in line with the desired false alarm rate when this is necessary.

Previous chapters have focused on methods for fitting accurate models to complex data. However, these monitoring approaches require data that is free of faults to fit an initial model. In practice, it can be difficult to obtain data that is fault free, even for the purposes of fitting a model for subsequent monitoring, and methods specifically designed to cope with outliers from the field of robust statistics are needed. In Chapter 6, a new approach for performing robust, sparse PCA called ROSPCA is introduced. ROSPCA is capable of unsupervised outlier identification, and variable selection which allows the practitioner to gain reliable insights into the structure of their process of interest. In the context of process monitoring, ROSPCA can be used to identify key variables for sensor deployment even when available data contains faults.

Chapter 7 concludes by summarizing the work and highlighting directions for further research.

A difficulty faced by practitioners wishing to perform SPM on high-dimensional processes is that software for even the basic PCA-based process monitoring methods is not available except for some implementations of specific methods in niche products. The appendix is a manual and introduction to a Matlab toolbox collecting the important PCA-based SPM methods, and well as a number of other multivariate SPM methods suitable for processes containing only a few variables. The toolbox allows for script based programming, but also provides GUI's for simulating data and performing process monitoring on data supplied by the user.

Chapter 2

Literature review of PCA-based monitoring methods

Based on: Hubert, M., De Ketelaere, B., Schmitt, E. (2015). Overview of PCA-based statistical process monitoring methods for time-dependent, high-dimensional data. *Journal of Quality Technology* 47 (4), 318–335.

2.1 Introduction

Quality control charts are a widely used tool, developed in the field of statistical process monitoring (SPM) to identify when a system is deviating from typical behavior. High-dimensional, time-dependent data frequently arise in applications ranging from health care, industry, IT, and economy. These data features challenge many canonical SPM methods, which lose precision as the dimensionality of the process grows, or are not well-suited for monitoring processes with a high degree of correlation between variables. In this paper, we present an overview of foundational principal component analysis-based techniques currently available to cope with these process types, and indicate some advantages and disadvantages. A wide range of scenarios encountered in SPM have motivated the development of many control chart techniques, which have been improved and reviewed over the course of the last forty years. Bersimis et al. (2006) give an overview of many multivariate process monitoring techniques, such

as the multivariate EWMA and multivariate CUSUM, but provides minimal coverage of techniques for high-dimensional processes. Barceló et al. (2010) compare the classical multivariate time series Box-Jenkins methodology with a partial least squares (PLS) method. The latter is capable of monitoring high-dimensional processes, but more methods for a broader range of time-dependent process scenarios are not covered. In discussing the monitoring of multivariate processes, Bisgaard (2012) highlights principal component analysis (PCA), partial least squares, factor analysis and canonical correlation analysis as applicable monitoring methods. These methods and their extensions have the property that they are capable of handling high-dimensional process data, and time-dependence. All of them project the high-dimensional process onto a lower dimensional subspace, and monitor the process behavior with respect to it. Woodall and Montgomery (2014) provide a survey of multivariate process monitoring techniques as well as motivations for their use. The authors also provide clear insights into possible process types and which monitoring methods might be suitable, and offer commentary on popular performance measures, such as the average run length and false discovery rate. Other books and papers devote more attention to PCA process monitoring. Kourti (2005) describes fundamental control charting procedures for latent variables, including PCA and PLS, but does not discuss many of the main methods for time-dependent data nor their extensions. Kruger and Xie (2012) includes a chapter covering the monitoring of high-dimensional, time-dependent processes, but focuses on one method only. Qin (2003) provides a review of fault detection, identification and reconstruction methods for PCA process monitoring. He mentions the challenges of monitoring time-dependent processes, but restricts his primary results to cases where the data is not time-dependent. However, to the best of our knowledge, an overview directly focusing on the range of available control chart techniques concerned with high-dimensional, time-dependent data has not yet been written with directions for practical use.

We assume that we have observed a large number, p, of time series $\boldsymbol{x}_j(t_i)$, $(1 \leqslant j \leqslant p)$ during a calibration period $t_1, t_2, \ldots, t_T$. As time continues, more measurements become available. SPM aims to detect deviations from typical process behavior during two distinct phases of process measurement; called Phase I, and Phase II. Phase I is the practice of retrospectively evaluating whether a previously completed process was statistically in control. Phase II is the practice of determining whether new observations from the process are in control as they are measured. Two types of time-dependence are autocorrelation, and non-stationarity. Autocorrelation arises when the measurements within one time series are not independent. Non-stationarity arises when the parameters governing a process, such as the mean or covariance, change over time. While it can be advantageous to include process knowledge, such as information about normal state changes, for the sake of focus we will assume no such prior

knowledge.

When no autocorrelation is present in the data, and the process is stationary, control charts based on PCA have been successfully applied in process monitoring settings with high-dimensionality. These methods operate by fitting a model on a $T \times p$ calibration data matrix $\boldsymbol{X}_{T,p}$, where the i-th row in the j-th column contains the i-th measurement of the j-th time series $\boldsymbol{x}_j(t_i)$ for $1 \leqslant i \leqslant T$. The number of rows of $\boldsymbol{X}_{T,p}$ thus refers to the number of observed time points, and the number of columns to the number of time-series measured in the system. The calibration data are chosen to be representative of typical behavior of the system. A new observation at time t, $\boldsymbol{x}(t) = (\boldsymbol{x}_1(t), \boldsymbol{x}_2(t), \ldots, \boldsymbol{x}_p(t))'$, is compared to the data in $\boldsymbol{X}_{T,p}$, and evaluated by the control chart to determine whether it is typical. This is called Static PCA because the fitted model remains static as new observations are obtained. Therefore, it will not adjust as underlying parameter values change (non-stationarity), and no attempt is made to model relationships between observations at different time points (autocorrelation). One can identify autocorrelation in a process by examining autocorrelation and cross-correlation functions of the data, as we shall do below. Non-stationarity can be assessed on univariate data using the augmented Dickey-Fuller test for a unit root. In high-dimensional data, a compromise is to perform this test on each of the scores of a Static PCA model.

Three classes of approaches have been proposed to extend PCA methods to cope with time-dependent data. These are Dynamic PCA (DPCA), Recursive PCA (RPCA), and Moving Window PCA (MWPCA). DPCA was developed to handle autocorrelation, whereas RPCA and MWPCA are able to cope with non-stationary data. No method is currently proposed for settings when both autocorrelation and non-stationarity are present. Although existing methods may provide acceptable monitoring in some contexts, this is nonetheless an area for further research.

2.2 Introducing the NASA bearings data set

Throughout this dissertation, the NASA Prognostics Center of Excellence Bearing data set [Lee et al. (2007)] will be used to illustrate the behavior of the methods on data with autocorrelation and non-stationarity. As shown in Figure 2.1, the data consist of measurements of eight sensors ($p = 8$), with each sensor representing either the x or y-axis vibration intensities of a bearing. Four bearings are monitored at intervals of approximately 15 minutes, and a vibration signal of about a second is recorded to describe the "stability." These raw data are then compressed into a single feature for each sensor. The resulting

observations are 8-dimensional vectors of bearing vibration intensities spaced at approximately 15 minute intervals. These are paired, such that the first two sensors correspond to the first bearing and so on. Figure 2.1 shows that there are two variables, belonging to the seventh and eighth sensors corresponding to the fourth bearing (plotted in light orange), which begin to deviate from typical behavior shortly after the 600th observation. Later in the experiment, a catastrophic failure for all of the bearings is observed.

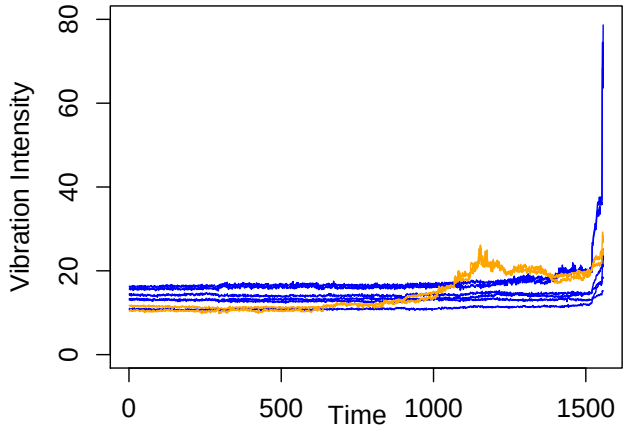

Figure 2.1: Data series depicting the autocorrelated, non-stationary NASA ball bearing data set. Sensors 7 and 8 are plotted in light orange. Other sensors are plotted in dark blue.

The NASA process shares many similarities with a multi-stream process (MSP). An MSP results in multiple streams of output for which, from the perspective of SPM, the quality variable and its specifications are identical across all streams. An MSP may also be defined as a continuous process where multiple measurements are made on a cross-section of the product [Epprecht et al. (2011)]. The NASA process has features of both of these definitions. It resembles the first in the sense that each of the bearings may be seen as having similar specifications to one another, with the average vibrations tending to be slightly different (but this can be adjusted so that they have the same mean), and the displayed variance being similar. The NASA process resembles the second definition in the sense that multiple measurements are made on a cross-section of the process; namely, all of the bearings are measured by two sensors. We detect some correlation between the streams, but as Epprecht and Simões (2013) note, this violates the assumption, made by most MSP methods, that none is present. Given these process features, PCA and its extensions are a possible monitoring solution. Runger et al. (1996) applied PCA to MSPs, and note that

this approach models the correlation structure between process variables. PCA is also capable of monitoring more general multivariate processes consisting of outputs that do not have identical properties, which may be the case when the second MSP definition is more appropriate, and multiple measurements are made on a cross-section. An additional advantage of PCA is that it is capable of modeling high-dimensional processes, which can pose problems for many MSP methods requiring an invertible covariance matrix.

Histograms, correlations, and pairwise scatterplots of vibration intensity measurements from sensors (1 and 2) placed on a typical bearing and sensors (7 and 8) on a deviating bearing are presented in Figure 2.2 for the first 120 observations, since these exhibit behavior characteristic of the in-control process. The corresponding autocorrelation functions (ACFs) up to fifty lags are depicted in Figure 2.3. The autocorrelation is presented as light-orange bars, while a limit to identify lags with high autocorrelation is expressed as a dark-blue line. During this early period, the pairs of sensors are only mildly correlated, with autocorrelation only exceeding the dark blue line indicating the 97.5 percentile limits for a few lags. For comparative purposes, the descriptive plots and autocorrelation functions are also shown for observations between $t = 600$ and $t = 1000$ in Figures 2.4 and 2.5. In the plots for the later time period, we see that sensors seven and eight become highly correlated as failure occurs. An advantage of multivariate control charts is that they take the change in the correlation between variables into account when determining if a system is going out of control. Furthermore, since non-stationarity has begun to develop, the ACFs now report very high order autocorrelation.

Earlier observations will be used to fit models, but control charts will also be used to assess these observations. In our context, we will consider this monitoring Phase I because it could be used by the practitioner to gain a better understanding of the behavior of this process from historical data. For the purposes of this paper, we will consider the later observations to be absent from the historical observations the practitioner could access for Phase I monitoring, and thus monitoring these later observations will constitute Phase II.

2.3 Static PCA

2.3.1 Method

principal component analysis defines a linear relationship between the original variables of a data set, mapping them to a set of uncorrelated variables. In general, Static PCA assumes to have observed an $(n \times p)$ data matrix $\boldsymbol{X}_{n,p} =$

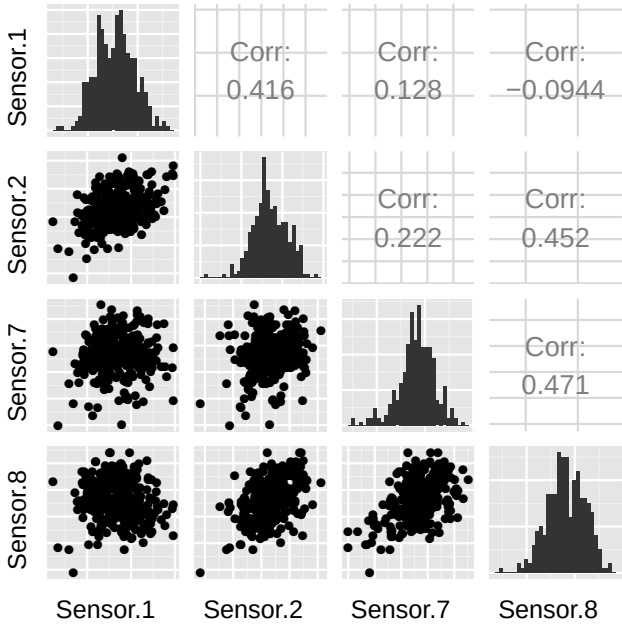

Figure 2.2: Histograms, scatterplots and correlations of sensors 1, 2, 7 and 8 during the first 120 measurements.

$(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)'$. Let $\mathbf{1}_n = (1, 1, \ldots, 1)'$ be of length n. Then the mean can be calculated as $\bar{\boldsymbol{x}} = \frac{1}{n}\boldsymbol{X}'_{n,p}\mathbf{1}_n$ and the covariance matrix as $\boldsymbol{S} = \frac{1}{n-1}(\boldsymbol{X}_{n,p} - \mathbf{1}_n\bar{\boldsymbol{x}}')'(\boldsymbol{X}_{n,p} - \mathbf{1}_n\bar{\boldsymbol{x}}')$. Each p-dimensional vector $\boldsymbol{x}$ is transformed into a score vector $\boldsymbol{y} = \boldsymbol{P}'(\boldsymbol{x} - \bar{\boldsymbol{x}})$ where $\boldsymbol{P}$ is the $p \times p$ loading matrix, containing columnwise the eigenvectors of $\boldsymbol{S}$. More precisely, $\boldsymbol{S}$ can be decomposed as $\boldsymbol{S} = \boldsymbol{P}\boldsymbol{\Lambda}\boldsymbol{P}'$. Here, $\boldsymbol{\Lambda} = \text{diag}(\lambda_1,\ \lambda_2, \ldots,\ \lambda_p)$ contains the eigenvalues of $\boldsymbol{S}$ in descending order. Throughout this paper, PCA calculations will be performed using the covariance matrix. However, it is generally the case that the methods discussed can also be performed using the correlation matrix $\boldsymbol{R}$ by employing different formulas.

It is common terminology to call $\boldsymbol{y}$ the *scores* and the eigenvectors, $\boldsymbol{P}$, the *loading* vectors. In many cases, due to redundancy between the variables, fewer components are sufficient to represent the data. Thus, using $k < p$ of the

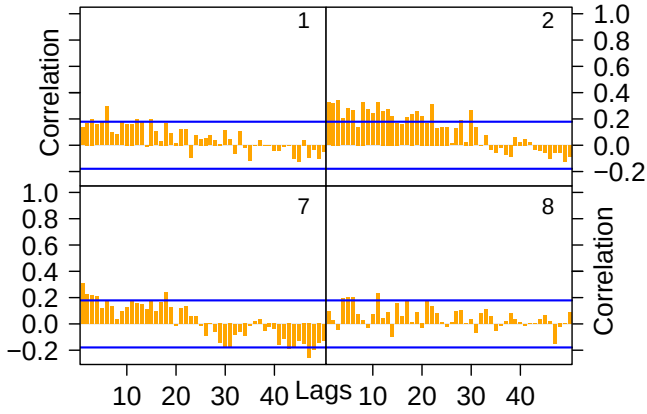

Figure 2.3: ACFs of sensors 1, 2, 7 and 8 during the first 120 measurements.

components, one can obtain k-dimensional scores by the following:

$$\boldsymbol{y} = \boldsymbol{P}_k'(\boldsymbol{x} - \bar{\boldsymbol{x}}) \tag{2.1}$$

where $\boldsymbol{P}_k$ contains only the first k columns of $\boldsymbol{P}$. To select the number of components to retain in the PCA model, one can resort to several methods, such as the scree plot or cross-validation. For a review of these, and other methods, see e.g. Valle et al. (1999) and Jolliffe (2002). In this paper, the number of components will be selected based on the cumulative percentage of variance (CPV), which is a measure of how much variation is captured by the first k PCs:

$$\text{CPV}(k) = \frac{\sum_{j=1}^{k} \lambda_j}{\sum_{j=1}^{p} \lambda_j} 100\%.$$

The number of PCs is selected such that the CPV is greater than the minimum amount of variation the model should explain.

Control charts can be generated from PCA models by using the *Hotelling's* T^2 statistic and the Q-statistic, which is also sometimes referred to as the *Squared Prediction Error* (SPE). One measure of the sensitivity of the control charts is their false discovery rate (FDR), which is the probability of classifying an in-control observation as out-of-control. Throughout this dissertation, target FDR will be 1% unless otherwise mentioned. This rate is still high for industrial applications, but the theory and results developed throughout the dissertation translate to more conservative FDRs.

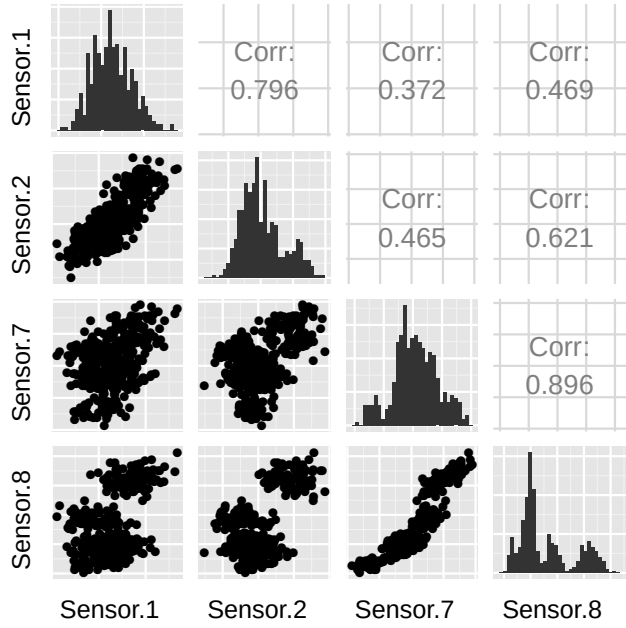

Figure 2.4: Histograms, scatterplots and correlations of sensors 1, 2, 7 and 8, during the time period between $t = 600$ and $t = 1000$.

For any p-dimensional vector $\boldsymbol{x}$ Hotelling's T^2 is defined as:

$$T^2 = (\boldsymbol{x} - \bar{\boldsymbol{x}})' \boldsymbol{P}_k \boldsymbol{\Lambda}_k^{-1} \boldsymbol{P}_k' (\boldsymbol{x} - \bar{\boldsymbol{x}}) = \boldsymbol{y}' \boldsymbol{\Lambda}_k^{-1} \boldsymbol{y}$$

where $\boldsymbol{\Lambda}_k = \text{diag}(\lambda_1,\ \lambda_2, \ldots,\ \lambda_k)$ is the diagonal matrix consisting of the k largest eigenvalues of $\boldsymbol{S}$. The Q-statistic is defined as:

$$Q = (\boldsymbol{x} - \bar{\boldsymbol{x}})' (\boldsymbol{I} - \boldsymbol{P}_k \boldsymbol{P}_k') (\boldsymbol{x} - \bar{\boldsymbol{x}}) = ||\boldsymbol{x} - \hat{\boldsymbol{x}}||^2$$

with $\hat{\boldsymbol{x}} = \boldsymbol{P_k} \boldsymbol{P}_k' (\boldsymbol{x} - \bar{\boldsymbol{x}})$. The Hotelling's T^2 is the Mahalanobis distance of $\boldsymbol{x}$ in the PCA model space, and the Q-statistic is the quadratic orthogonal distance to the PCA space. Assuming temporal independence and multivariate normality of the scores, the $100(1-\alpha)\%$ control limit for Hotelling's T^2 is

$$T_\alpha^2 = \frac{k(n^2-1)}{n(n-k)} F_{k,n-k}(\alpha). \tag{2.2}$$

Here, $F_{k,n-k}(\alpha)$ is the $(1-\alpha)$ percentile of the F-distribution with k and $n-k$ degrees of freedom. If the number of observations is large, the control limits

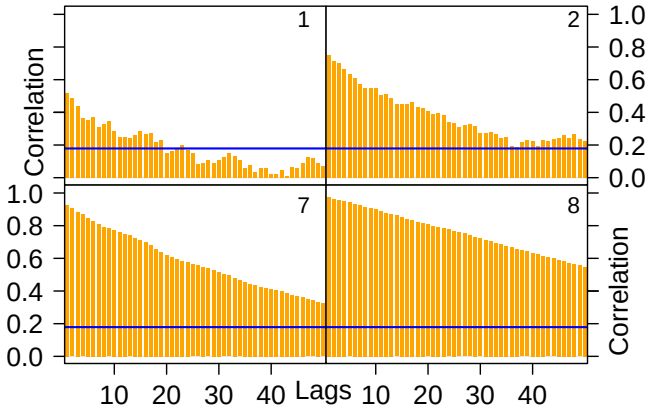

Figure 2.5: ACFs of sensors 1, 2, 7 and 8 during the time period between $t = 600$ and $t = 1000$.

can be approximated using the $(1 - \alpha)$ percentile of the χ^2 distribution with k degrees of freedom, thus $T^2_\alpha \approx \chi^2_k(\alpha)$. The simplicity of calculating this limit is advantageous. The control limit corresponding to the $(1 - \alpha)$ percentile of the Q-statistic can be calculated, provided that all the eigenvalues of the matrix $\boldsymbol{S}$ can be obtained [Jackson and Mudholkar (1979)]:

$$Q_\alpha = \theta_1 \left(\frac{z_\alpha \sqrt{2\theta_2 h_0^2}}{\theta_1} + 1 + \frac{\theta_2 h_0 (1 - h_0)}{\theta_1^2} \right)^{1/h_0}$$

where

$$\theta_i = \sum_{j=k+1}^{p} \lambda_j^i \quad \text{for} \quad i = 1, 2, 3 \quad \text{and} \quad h_0 = 1 - \frac{2\theta_1 \theta_3}{3\theta_2^2}$$

and z_α is the $(1 - \alpha)$ percentile of the standard normal distribution. Another way of obtaining cut-offs for the Q-statistic based on a weighted χ^2 distribution is detailed in Nomikos and MacGregor (1995). An advantage of this approach is that it is relatively fast to compute. During Phase I the T^2 and Q-statistic are monitored for all observations $\boldsymbol{x}(t_i) = (\boldsymbol{x}_1(t_i), \ldots, \boldsymbol{x}_p(t_i))'$ with $1 \leqslant i \leqslant T$. It is important to note that fitting the PCA model to this data will result in a biased model with possible inaccurate fault detection if faults are present, since they can bias the fit. If faults are present, it is advised to fit a robust PCA model and refer to the monitoring statistics it produces. Phase II consists of

evaluating contemporary observations $\boldsymbol{x}_t = \boldsymbol{x}(t)$ using the T^2 and Q statistic based on the outlier-free calibration set.

The most commonly used approximation for the control limit of the Q-statistic is given by Jackson and Mudholkar (1979). This approximation typically performs well, but strongly relies on the assumption that the $k+1, \ldots, p$ eigenvalues are small. This assumption may be violated in the presence of high autocorrelation (especially when applying an adaptive method, such as RPCA or MWPCA), if faulty observations enter the calculation of the updated covariance matrix. If this occurs, the limits can become uninformative. To circumvent this issue we resort to the general result of Box (1954), which shows that the Q-statistic is approximately distributed as a scaled χ^2-distribution with h degrees of freedom, denoted as $g\chi^2_h$. Provided that all the eigenvalues of $\boldsymbol{S}$ are available, the parameters are given by:

$$\theta_i = \sum_{j=k+1}^{p} \lambda_j^i \text{ for } i = 1, 2;\ g = \frac{\theta_2}{\theta_1};\ \text{and } h = \frac{\theta_1^2}{\theta_2}.$$

The control limit for the Q-statistic, Q_α, is then taken as the $(1-\alpha)$ quantile of the $g\chi^2_h$ distribution. We compare the difference in performance between the two limit derivations in Figure 2.6 for RPCA on an AR(1) process (defined in Chapter 3, Equation 3.3. Here, we see that after the introduction of a fault at $t = 500$ the Box (1954) approximation continues to produce realistic limits, whereas the Jackson and Mudholkar (1979) limit drops to an unrealistically low value as a result of faulty observations contaminating the covariance matrix.

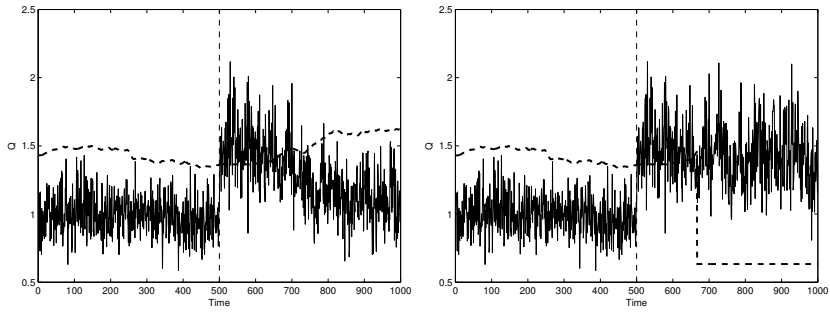

Figure 2.6: RPCA Q-statistic control charts for the Box (left) and Jackson and Mudholkar (right) limits on AR(1) data with a score fault at $t = 500$.

An intuitive depiction of Static PCA is given in Figure 2.7. This figure will serve as a basis of comparison between the DPCA, RPCA and MWPCA techniques that are discussed in the following sections. Variables are represented as vertical

lines of dots measured over time. The light-red rectangle contains the observed data during the calibration period that is used to estimate the model that will be used for subsequent monitoring. The dark-blue rectangle is the new observation to be evaluated. The two plots show that at time $t + 1$ (right) the same model is used to evaluate the new observation in dark blue as in the previous time period, t (left).

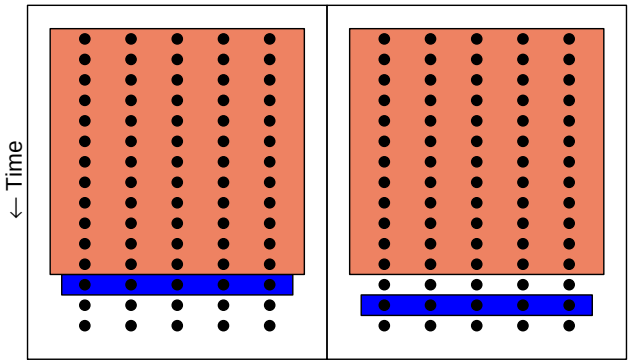

Figure 2.7: A schematic representation of Static PCA at times t (left) and $t + 1$ (right). The model is fitted on observations highlighted in light-red. The new observation, highlighted in dark-blue, is evaluated.

PCA is well suited for monitoring processes where the total quality of the output is properly assessed by considering the correlation between all variables. However, if a response variable is also measured and the relationship of the process variables to it is of primary interest, the technique of partial least squares (PLS) is preferred to PCA. Like linear regression, it is used to model the linear relation between a set of regressors and a set of response variables, but like PCA it projects the observed variables onto a new space, allowing it to cope with high-dimensional data. Control charts may be implemented for PLS, in much the same way as they are for PCA. Kourti (2005) provides a comparison of PCA and PLS, as well as some references for PLS control chart literature.

Static PCA requires a calibration period to fit a model. However, PCA is highly susceptible to outliers, which exert disproportionate influence on the classical covariance matrix estimate. If outliers are included in the data used to fit a monitoring model, the detection accuracy can be severely impaired. Robust PCA methods, such as ROBPCA Hubert et al. (2005), have been designed to provide accurate PCA models even when outliers are present in the data. A

robust PCA method can be used to identify outliers in the calibration data for removal or examination. Once these are removed, the resulting robust PCA model can be used as the basis for subsequent process monitoring. ROBPCA may be performed using the `robpca` function in the `LIBRA` toolbox [Verboven and Hubert (2005)], or the `PcaHubert` function in the `R` package `rrcov` [Todorov and Filzmoser (2009)].

In addition to outliers, future observations with missing data and observations with missing data during the calibration phase present challenges for process monitoring. In the context of PCA control charts, a number of options for addressing these issues exist. The problem of future observations with missing data is typically addressed by using the process model and non-missing elements of the new observation, $\boldsymbol{x}_{new}$, to correct for the missingness of some of its elements. Examples of algorithms using this approach at various levels of complexity are discussed in Arteaga and Ferrer (2002). They conclude that a method referred to as trimmed score regression (TSR) has the most advantages, in terms of accuracy and computational feasibility, of the methods they considered. TSR uses information from the full score matrix $\boldsymbol{Y}$ from the calibration data, the loadings in $\boldsymbol{P}$ corresponding to the non-missing variables in $\boldsymbol{x}_{new}$ and $\boldsymbol{x}_{new}$ itself, to estimate the $\boldsymbol{y}_{new}$. In the event that the calibration data has missing values, one does not have access to existing estimates of $\boldsymbol{P}$ and $\boldsymbol{Y}$ to use for missing data corrections. Walczak and Massart (2001) propose a method for missing data imputation based on the expectation maximization (EM) algorithm. Serneels and Verdonck (2008) make this method robust, allowing missing data imputation to proceed even when the calibration data set is contaminated by outliers. An implementation is available in the `rrcovNA` package [Todorov (2013)] in R [R Core Team (2014)]. Folch-Fortuny et al. (2015) further investigated missingness in PCA and indicate a number of methods that can deliver better performance than the EM algorithm.

2.3.2 Static PCA applied to the NASA data

In this subsection we apply Static PCA to the NASA data. Before constructing control charts, we performed ROBPCA on the first 120 observations that we use to fit the PCA model. No significant outliers were detected, so we fit a PCA model on that data without removing observations. No data was missing in this data set, so missing data methods were not employed. It is common in many fields to perform preprocessing. The type of preprocessing is typically determined by the type of process being monitored, with chemometrics, for instance, giving rise to many preprocessing approaches specific to that context. In the case of the NASA data, no special preprocessing is necessary. Since all of the sensors are measuring vibration in the same units, standardizing the data is

not strictly necessary, but it will be performed for all of the methods considered since the adaptive methods will perform it automatically.

Static PCA applied to the NASA bearing data set generates the control chart in Figure 2.8 and ACF plot in Figure 2.9. We plot the logarithm of the T^2 and Q-statistics in these and subsequent charts as solid light-orange lines, and the control limit in solid, dark-blue lines. The first 120 observations are used to fit the underlying model, as we do not observe any large change in the vibration intensity of any of the sensors during this period, and this will also allow us to evaluate the estimated model against the well-behaved data observed before $t = 120$. Therefore, we differentiate between Phase I, which takes place when $t \leqslant 120$ and Phase II. A vertical line divides these two periods in Figure 2.8. Five components are retained in accordance with the CPV criterion. We see that failure of the system is detected before catastrophic failure occurs, at around $t = 120$ by the Q-statistic, and at around $t = 300$ by the T^2-statistic. Since we did not detect any major outliers using ROBPCA during Phase I, it is not surprising that few observations exceed the cut-offs during this early period and that later during Phase II when the issue with the fourth bearing develops we find a failure. Figure 2.9 shows there is room to reduce the variability of the statistics by accounting for autocorrelation. Examining the first score, we see that the autocorrelations are fairly low, but when the number of lags is less than ten or more than thirty, many exceed the cutoff. The second component exhibits even stronger autocorrelation. Reducing the autocorrelation will more strongly justify the assumption that the control chart statistics are being calculated on i.i.d. inputs.

It is desirable that a model of the data be interpretable [Camacho et al. (2010)]. One way to interpret PCA is by examining the loadings it produces. In some cases, this reveals a logical structure to the data. Table 2.1 presents the loadings of the Static PCA model of the NASA data. In the case of this data set, a clear structure is not revealed by the loadings. The first component loads most heavily on sensors 1, 2, and 5. It is understandable that the sensors 1 and 2 might be correlated since they both measure the first bearing, but sensor 5 measures the third bearing. The remaining components are similarly ambiguous, with none corresponding to an intuitive structure. One way to improve interpretabilty of PCA models is to employ a rotation, such as the varimax. However, doing so is not necessary to achieve desirable fault detection properties. The last three components differ from the first two in that some of the values of the loadings are so small that they are effectively zero (these are left blank in the table). The omission of relatively unimportant variables from components increases the interpretability of them. Two similar procedures for accomplishing this are Sparse PCA [Zou et al. (2006)] and SCoTLASS [Jolliffe et al. (2003)]. These methods are designed to return a PCA model which fits the data well, while

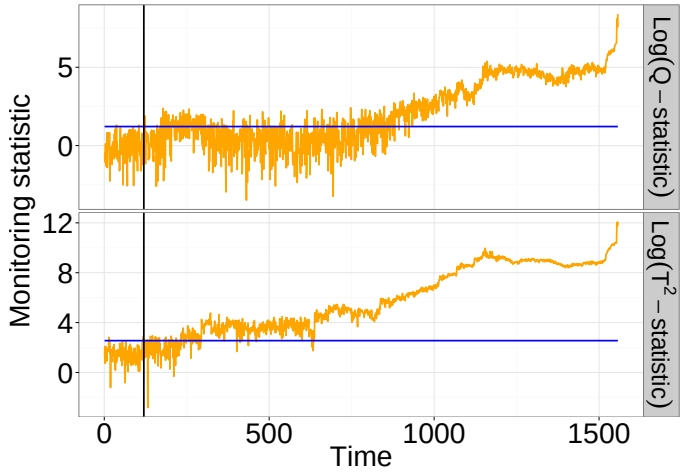

Figure 2.8: Static PCA control charts for the entire NASA data set. The first 120 observations are used to fit the underlying model.

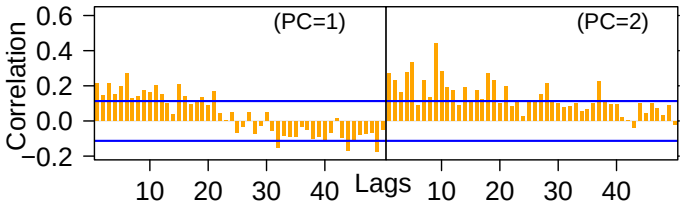

Figure 2.9: ACFs of the first two scores of Static PCA applied to the NASA data set for $t \leqslant 120$.

giving many variables small or zero loadings on the components where they are relatively unimportant.

As a byproduct of PCA, one can construct a contribution plot, showing the contribution of each variable to the control statistics for a given observation [Miller et al. (1998)]. The contributions of the jth variable to the T^2 and the Q-statistic of an observation $\boldsymbol{x}$ is the jth element of the vectors:

$$T^2_{contr} = (\boldsymbol{x} - \bar{\boldsymbol{x}})' \boldsymbol{P}_k \boldsymbol{\Lambda}_k^{-1/2} \boldsymbol{P}_k' \tag{2.3}$$

$$Q_{contr} = (\boldsymbol{x} - \bar{\boldsymbol{x}})' (\boldsymbol{I} - \boldsymbol{P}_k \boldsymbol{P}_k').$$

Table 2.1: Loadings of the Static PCA model of the NASA data.

		Component				
		1	2	3	4	5
Sensor	1	-0.471	0.231	-0.173	0.264	
	2	-0.430	0.306	-0.341	0.403	
	3	-0.249	0.175	-0.194	-0.400	-0.359
	4	-0.259	0.110	-0.320	-0.570	
	5	-0.467	-0.615	0.205	0.301	-0.240
	6	-0.368	-0.464		-0.418	0.269
	7	-0.236	0.422	0.788	-0.128	-0.276
	8	-0.233	0.198	0.212		0.807

These contributions can be plotted as bars with the expectation that variables which made a large contribution to a fault can be identified by higher magnitude bars. This does not necessarily lead to precise identification of the source of the fault, but it shows which variables are also behaving atypically at the time of occurrence. In Figure 2.10, we display contribution plots for observations before the fault ($t = 100$) and after ($t = 1200$). Comparing the two plots, we see that both statistics are much less influenced by the observation from $t = 100$ than from $t = 1200$. Focusing on the contribution plots for later observation, we see that the plot for the Q-statistic is ambiguous, but that the contribution plot for the T^2-statistic clearly indicates sensors 7 and 8 as the primary sources for this observation's deviation on the model space. Interpreting these plots, the practitioner would likely investigate the fourth bearing more closely. When many variables are being monitored, the contribution plot can become difficult to interpret. Hierarchical contribution plots are a way of overcoming this issue [Qin et al. (2001)]. Qin (2003) provide further detail on extensions to the contribution plot and fault reconstruction. Other approaches to interpreting PCA models have been examined in Camacho et al. (2010).

2.4 Dynamic PCA

One approach for addressing autocorrelation is to perform first-order differencing. This can diminish the effects of autocorrelation, but it is problematic in the context of process monitoring. Problems arise when detection of some fault types, such as step faults, is desired. In the case of step faults, differencing will reveal the large change that takes place when the fault first occurs, but subsequent faulty observations will appear normal since they are in control

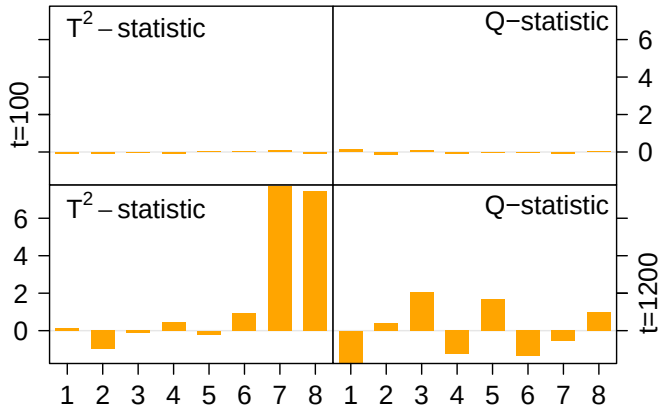

Figure 2.10: Contribution plots showing the contribution of each sensor to the T^2 and Q-statistics for observations at $t = 100$ and $t = 1200$.

relative to one another. As a result, an operator interpreting the control chart may be led to believe that the first faulty observation was an outlier, and the process is back in control. Dynamic PCA was first proposed in Ku et al. (1995) as a way to extend Static PCA tools to autocorrelated, multivariate systems. The authors note that previously, others had taken the approach of addressing autocorrelated data by fitting univariate autoregressive integrated, moving average (ARIMA) models to the data and analyzing the residuals which ignores cross-correlation between the variables. Attempts were made to improve the results by estimating multivariate models using this approach, but this quickly proves to be a complex task as p grows, due to the high number of parameters that must be estimated and the presence of cross-correlation.

DPCA combines the facility in high dimensions of PCA with the ability to cope with autocorrelation of ARIMA. The approach of Ku et al. (1995) is that in addition to the observed variables, the respective lagged values up to the proper order can also be included as input for PCA estimation. For example, an AR(1) process will require the inclusion of lagged values up to order one.

Given data observed up to time T, $\boldsymbol{X}_{T,p}$, DPCA with one lag models the process based on a matrix including one lag, $\widetilde{\boldsymbol{X}}_{T-1,2p}$, which has twice as many variables and one fewer row as a result of the lagging. More generally for an AR(l) process, we obtain $\widetilde{\boldsymbol{X}}_{T-l,(l+1)p}$, where the ith row of $\widetilde{\boldsymbol{X}}_{T-l,(l+1)p}$ is $(\boldsymbol{x}(t_{i+l}), \boldsymbol{x}(t_{i+l-1}), \ldots, \boldsymbol{x}(t_i))$ with $i = 1, \ldots, T - l$. As new observations are measured, they are also augmented with lags as in the rows of $\widetilde{\boldsymbol{X}}_{T-l,(l+1)p}$, and compared to the model estimated by DPCA. In estimating the linear

relationships for the dimensionality reduction, this method also implicitly estimates the autoregressive structure of the data, as e.g. illustrated in Tsung (2000). For addressing the issue of moving average (MA) terms, it is well known that an MA process can be approximated by using a high enough order AR process. As functions of the model, the T^2 and Q-statistics now will also be functions of the lag parameters. If the outlier detection methods discussed in Section 2.3 are of interest, they can be applied after including the appropriate lags.

DPCA is characterized intuitively in Figure 2.11, where a model estimated from observations in the light-red window is used to evaluate whether the newly observed observation and the corresponding lagged observations, in dark-blue, deviate from typical behavior. Note that because the assumption is that the mean and covariance structures remain constant, it is sufficient to use the same model to evaluate observations at any future time point.

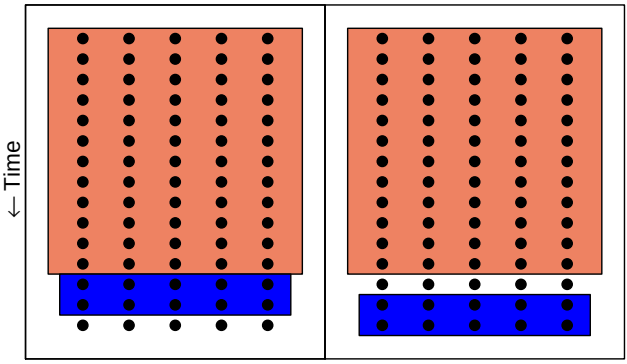

Figure 2.11: A schematic representation of DPCA with one lag at times t (left) and $t + 1$ (right).

Ku et al. (1995) demonstrate that their procedure accounts for the dynamic structure in the raw data, but note that the score variables will still be autocorrelated and possibly cross-correlated, even when no autocorrelation is present. Kruger et al. (2004) prove the scores of DPCA will inevitably exhibit some autocorrelation. They show that the presence of autocorrelated score variables leads to an increased rate of false alarms from DPCA procedures using Hotelling's T^2. They claim that the Q-statistic, on the other hand, is applied on the model residuals, which are assumed to be i.i.d., and thus this statistic is not affected by autocorrelation of the scores. They propose to remedy the presence of autocorrelation in the scores through ARMA filtering. Such an ARMA filter

can be inverted and applied to the score variables so that unautocorrelated residuals are produced for testing purposes. Another possibility is to apply an ARMA filter on the process data, but in cases where the data is high-dimensional, it is generally more practical to work on the lower-dimensional scores.

Luo et al. (1999) propose that the number of false alarms generated using DPCA methods can be reduced by applying wavelet filtering to isolate the effects of noise and process changes from the effects of physical changes in the sensor itself. This approach does not specifically address problems of autocorrelations and non-stationarity, but the authors find that results improve when a DPCA model is applied to autocorrelated data that has been filtered.

Another approach to reduce the autocorrelation of the scores was introduced and explored by Rato and Reis (2013a) and Rato and Reis (2013c). Their method DPCA-DR proceeds by comparing the one-step ahead prediction scores (computed by means of the Expectation-Maximization algorithm) with the observed scores. The resulting residuals are almost entirely uncorrelated, and therefore suitable for monitoring. Statistics based on this approach are typically better behaved than those produced by both Static and conventional DPCA, sometimes significantly so.

2.4.1 Choice of parameters

A simple way to select the number of lags manually is to apply a PCA model with no lags and examine the ACFs of the scores. If autocorrelation is observed, then an additional lag can be added. This process can be repeated until enough lags have been added to sufficiently reduce the autocorrelation. However, this approach is extremely cumbersome due to the number of lags that it may be necessary to investigate, and similarly if there are many components, there will be many ACFs to inspect. Ku et al. (1995) provide an algorithm to specify the number of lags which follows from the argument that a lag should be included if it adds an important linear relationship. Beginning from no lags, their algorithm sequentially increases the number of lags and evaluates whether the new lag leads to an important linear relationship for one of the variables. This method explicitly counts the number of linear relationships. When a new lag does not reveal an important linear relationship, the algorithm stops and the number of lags from the previous iteration is used. The number of lags selected is usually one or two and all variables are given the same number of lags.

Rato and Reis (2013b) propose two new, complementary methods for specifying the lag structure. The first is a more robust method of selecting the common number of lags applied to all variables than the Ku et al. (1995) approach. It also increasingly adds lags, but the algorithm stops after l lags, if, roughly

said, the smallest singular value of the covariance matrix of the extended data matrix $\tilde{\boldsymbol{X}}$ is significantly lower than the one using $l-1$ lags. Intuitively, this corresponds to the new lag not providing additional modeling power. The second method begins from the previous one, and improves it by also reducing the number of lags for variables which do not require so many, thereby giving a variable determined lag structure. The authors show that this better controls for autocorrelation in the data, and leads to better behaviors of the test statistics.

2.4.2 DPCA applied to the NASA data

DPCA control charts for the NASA data are shown in Figure 2.12. Parameter values for DPCA and the adaptive methods are presented in Table 2.2. For DPCA, this is the number of lags; for RPCA, the forgetting factor η; and for MWPCA, the windowsize H. All models select the number of latent variables (LV) such that the CPV is at least 80%. The number of components used at the last evaluation of the system is included for each setting. Typically, the number of latent variables varies at the beginning of the control chart and then stabilizes to the value that is shown.

Table 2.2: Parameter values (PV) used in the NASA data example for all time-dependent methods.

	Low		High	
Method	LV	PV	LV	PV
DPCA	8	1	39	20
RPCA	2	0.9	2	0.9999
MWPCA	1	40	1	80

Proposals for automatically selecting the parameter of each of the methods are available, but a consensus does not exist on which is best for any of the three. Thus, for each method, we select low and high values for the parameter of interest to illustrate how this influences the performance. Nonetheless, we still note that automatic methods, such as those discussed for selecting the number of lags for DPCA, should be considered within the context facing the practitioner.

When DPCA is applied, the number of components needed to explain the structure of the model input grows. For one lag, 8 components are needed, while for 20 lags 39 components are taken. This has the shortcoming that data sets with few observations may not be able to support such a complex structure. Figure 2.12 shows the results of DPCA control charts fitted on the first 120

observations. Again, we consider the period when $t \leqslant 120$ as Phase I monitoring, and at later points Phase II monitoring takes place. When $l = 1$, the ACF of the first score (see Figure 2.13) exhibits autocorrelation at lags below ten and above twenty, as we saw in the case of Static PCA (see Figure 2.9). The second score of Static PCA showed autocorrelations exceeding the cut-off for almost all lags, but we now see that almost none exceed the cut-off. However, when 20 lags are used, we notice that in the right plot of Figure 2.12 the monitoring statistics are clearly autocorrelated. The ACFs of the first two scores, shown in Figure 2.13, confirm that autocorrelation is a major problem. This is an illustration of the trade-off between adding lags to manage autocorrelation and the issue that simply adding more can actually increase autocorrelation. A choice of the number of lags between 1 and 20 shows the progression towards greater autocorrelation.

It is possible to apply a contribution plot to a DPCA model, as we did for Static PCA. However, DPCA tends to use many more variables due to the inclusion of lags. This can make interpretation more difficult. A subspace approach for autocorrelated processes, such as the one proposed by Treasure et al. (2004), may be used to increase interpretability, though the authors note that the detection performance remains comparable to that of DPCA.

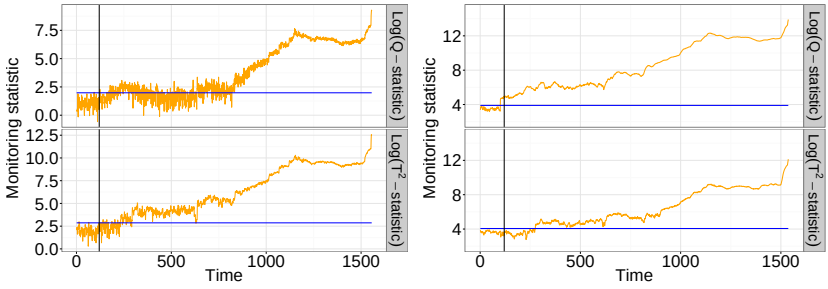

Figure 2.12: DPCA control charts for the NASA data set using 1 (left) and 20 (right) lags.

2.5 Recursive PCA

2.5.1 Method

Besides being sensitive to autocorrelation and moving average processes, Static PCA control charts are also unable to cope with non-stationarity. If a Static

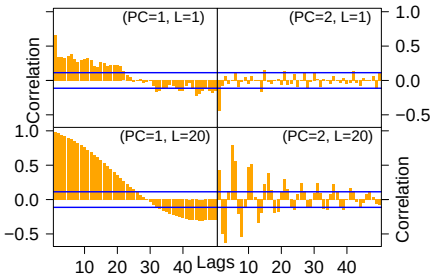

Figure 2.13: ACFs of the first two scores of DPCA applied to the NASA data set when using 1 (upper) and 20 (lower) lags for $t \leqslant 120$.

PCA model is applied to data with a non-stationary process in it, then issues can arise where the mean and/or covariance structure of the model become misspecified because they are estimated using observations from a time period with little similarity to the one being monitored. DPCA provides a tool for addressing autoregressive and moving average structures in the data. However, it is vulnerable to non-stationarity for the same reason as Static PCA. Differencing is a possible strategy for coping with non-stationarity, but it suffers from the same shortcoming as in the situation when the data is autocorrelated (see Section 2.4). In response to the need for an effective means of coping with non-stationarity, two approaches have been proposed: RPCA, and MWPCA. Both of these attempt to address non-stationarity by limiting the influence of older observations on estimates of the mean and covariance structures used to assess the status of observations at the most recent time point.

The idea of using new observations and exponentially downweighting old ones to calculate the mean and covariance matrix obtained from PCA was first investigated by Wold (1994) and Gallagher et al. (1997). However, both of these approaches require all of the historical observations and complete recalculation of the parameters at each time point. A more efficient updating approach was proposed in Li et al. (2000), which provided a more detailed treatment of the basic approach to mean and covariance/correlation updating that is used in the recent RPCA literature. A new observation is evaluated when it is obtained. If the T^2 or Q statistics exceed the limits because the observation is a fault or an outlier, then the model is not updated. However, when the observation is in control, it is desirable to update the estimated mean and covariance/correlation from the previous period. The approach of Li et al. (2000) was inspired by a recursive version of PLS by Dayal and MacGregor (1997b). This RPLS algorithm is supported by a code implementation in the counterpart paper [Dayal and MacGregor (1997a)].

More precisely, assume that the mean and covariance of all observations up to time t have been estimated by $\bar{\boldsymbol{x}}_t$, and $\boldsymbol{S}_t$. Then at time $t+1$ the T^2 and Q-statistic are evaluated in the new observation $\boldsymbol{x}_{t+1} = \boldsymbol{x}(t+1) = (\boldsymbol{x}_1(t+1), \ldots, \boldsymbol{x}_p(t+1))'$. If both values do not exceed their cut-off value, one could augment the data matrix $\boldsymbol{X}_{t,p}$ with observation $\boldsymbol{x}_{t+1}$ as $\boldsymbol{X}_{t+1,p} = [\boldsymbol{X}'_{t,p}\ \boldsymbol{x}_{t+1}]'$ and recompute the model parameters while using a forgetting factor $0 \leqslant \eta \leqslant 1$. In practice, updating is not performed using the full data matrix, but rather a weighting is performed to update only the parameters. Denoting n_t as the total number of observations measured at time t, the updated mean is defined as:

$$\bar{\boldsymbol{x}}_{t+1} = (1 - \frac{n_t}{n_t + 1}\eta)\boldsymbol{x}_{t+1} + \frac{n_t}{n_t + 1}\eta\ \bar{\boldsymbol{x}}_t,$$

and the updated covariance matrix is defined as:

$$\boldsymbol{S}_{t+1} = (1 - \frac{n_t}{n_t + 1}\eta)(\boldsymbol{x}_{t+1} - \bar{\boldsymbol{x}}_{t+1})(\boldsymbol{x}_{t+1} - \bar{\boldsymbol{x}}_{t+1})' + \frac{n_t}{n_t + 1}\eta\ \boldsymbol{S}_t.$$

This is equivalent to computing a weighted mean and covariance of $\boldsymbol{X}_{t+1,p}$, where older values are downweighted exponentially as in a geometric progression. Using a forgetting factor $\eta < 1$ allows RPCA to automatically give lower weight to older observations. As $\eta \to 1$, the model forgets older observations more slowly. The eigenvalues of $\boldsymbol{S}_{t+1}$ are used to obtain a loading matrix $\boldsymbol{P}_{t+1}$. Calculating the new loading matrix can be done in a number of ways that we touch upon when discussing computational complexity. Updating with correlation matrixes involves similar intuition, but different formulas. In order to lower the computational burden of repeatedly updating the mean and covariances, one strategy has been to reduce the number of updates, see He and Yang (2008). Application of the outlier detection and missing data methods discussed in Section 2.3 is problematic in the case of RPCA since those techniques are based on Static PCA and the number of observations used to initialize RPCA may be too short to apply them reliably. However, if the calibration data is assumed to be a locally stationary realization of the process, then it may be possible to apply them. These integration of such methods into adaptive PCA monitoring methods remains an open field in the literature.

RPCA is characterized intuitively in Figure 2.14, where a model estimated from observations in the light-red region is used to evaluate whether the newly observed observation, in dark blue, deviates from typical behavior. In this characterization, observations in the light-red region are given diminishing weight by a forgetting factor to reflect the relative importance of contemporary information in establishing the basis for typical behavior. As the choice of the forgetting factor varies, so does the weighting. Furthermore, new observations are later used to evaluate future observations because under the assumption

that the monitored process is non-stationary, new data is needed to keep the model contemporary. When an observation is determined to be out-of-control based on the T^2 or Q-statistic, then the model is not updated.

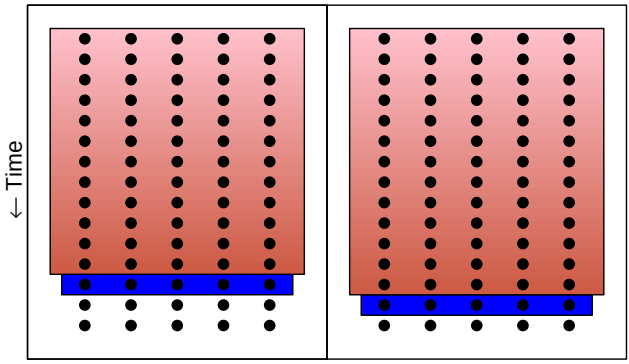

Figure 2.14: A schematic representation of Recursive PCA with a forgetting factor $\eta < 1$, at times t (left) and $t+1$ (right). The observations used to fit the model are assigned lower weight if they are older. This is represented by the lightening of the light-red region as the observations it covers become relatively old.

Updating the control limits is necessary as the dimensionality of the data could vary, and the underlying mean and covariance parameters of the PCA model change. In order to do so for the T^2, it is only necessary to recalculate $T^2_\alpha = \chi^2_{k_t}(\alpha)$ for the newly determined number of PCs, k_t. Furthermore, since $Q(\alpha)$ is a function of θ_i which are in turn functions of the eigenvalues of the covariance matrix, once the new PCA model has been estimated, the Q-statistic control limit is updated to reflect changes to these estimates. This is illustrated in the top (and bottom) plots of Figure 2.15, which shows RPCA control charts of the NASA data for low and high values of the forgetting parameter η. Here, we see that the cut-off of the T^2-statistic experiences small, sharp steps up as the number of components increases and down if they decrease. This is also the case for the cut-off of the Q-statistic, although the fluctuations are the result of the combined effects of a change in the number of components and the covariance structure of the data. The time at which the major fault is detected is clearly visible in the chart of the Q-statistic as the time point at which the control limit stops changing from $t = 637$.

In order to differentiate between outlier observations and false alarms, a rule is often imposed that a number of consecutive observations must exceed the

control limits before an observation is considered a fault (often 3 is used). Choi et al. (2006) propose that an effective way of using observations which may be outliers, or may prove to be faults is to implement a robust reweighting approach. Thus, when an observation exceeds the control limit, but is not yet determined to be a true fault in the process, they propose to use a reweighted version of the observed vector $\boldsymbol{x}$, where each component of $\boldsymbol{x}$ is downweighted according to its residual to the current model. The intention of this approach is to prevent outliers from influencing the updating process, while still retaining information from them instead of completely discarding them.

2.5.2 Choice of parameters

Selecting a suitable forgetting factor in RPCA is crucial. Typically, $0.9 \leqslant \eta \leqslant 0.9999$ since forgetting occurs exponentially, but lower values may be necessary for highly non-stationary processes. In Choi et al. (2006), RPCA is augmented using variable forgetting factors for the mean and the covariance or correlation matrix. This allows the model to adjust the rate of forgetting to suit a process with non-stationary. First, they define minimum and maximum values of the forgetting factors that can be applied to the mean and covariance, respectively. Then, they allow the forgetting factor to vary within those bounds based on how much the parameter has changed since the previous period relative to how much it typically changes between periods.

Computational complexity is an important concern faced by algorithms which perform frequent updates. Updating the mean is relatively straightforward, since doing so is only a rank-one modification. Updating the covariance matrix and then calculating the new loading matrix proves to be more involved. It is possible to proceed using the standard SVD calculation, but this is relatively slow, with $O(p^3)$ time, and hence other approaches to the eigen decomposition have been proposed. Kruger and Xie (2012) highlight the first order perturbation $[O(p^2)]$ and data projection method $[O(pk^2)]$ as particularly economical. When p grows larger than k, the data projection approach becomes faster relative to first order perturbations. However, the data projection approach assumes a constant value of k, and this is not a requirement of the first order perturbation method. When updating is performed in blocks, fewer updates are performed for a given period of monitoring which in turn reduces the computational cost.

2.5.3 RPCA applied to the NASA data

We apply two RPCA models to the NASA data. The first has a relatively fast forgetting factor of 0.9. This implies that it quickly forgets observations and

provides a more local model of the data than our second specification, which uses a slow forgetting factor of 0.9999. Both are initiated using a Static PCA model fitted on the first 120 observations, which according to our exploration of the NASA data, are stationary. Then, we apply the updating RPCA model to those data to obtain Phase I results. In this sense, Phase I serves as a validation set that the model is capable of monitoring the process when it is in control without producing a high false detection rate. We then proceed to apply the model to the observations after $t = 120$ constituting Phase II. We note that in practice, if the initializing period cannot be assumed stationary, then a fitting/validation approach based on continuous sets of data should be used to fit the model, with the validation set serving to prevent overfitting. Results for these two monitoring models are shown in Figure 2.15. Since the model with $\eta = 0.9$ (left) is based on a small set of observations, it is more local, but also less stable. This translates into a control chart with many violations of the control limit. Both the T^2 and Q-statistics detect failure before the end of the calibration period. In contrast, the model with $\eta = 0.9999$ (bottom) detects the failure at about $t = 600$ using the Q-statistic, and $t = 300$ using the T^2-statistic. The times of these detections are later than for Static PCA and DPCA because the RPCA model with $\eta = 0.9999$ is stable enough to produce a reliable model of the process, but adaptive enough that it adjusts to the increasingly atypical behavior of the fourth bearing during the early stages of its failure. This increased time to detecting the failure is a shortcoming of RPCA in this context, but the results also illustrate how it is capable of adapting to changes in the system. If these changes are natural and moderate, such adaptation may be desirable. Fault identification techniques are compatible with PCA methods for non-stationary data. The only restriction is that the model used for monitoring at the time of the fault should be the one used to form the basis of the contribution plot.

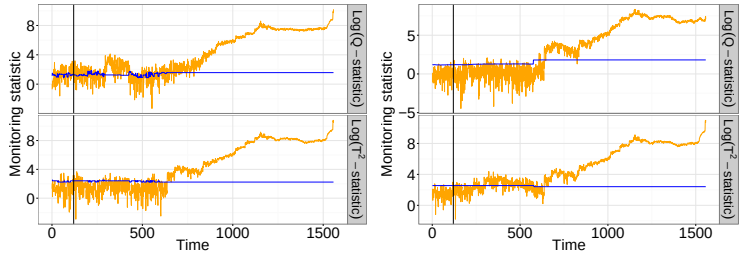

Figure 2.15: RPCA control charts for the NASA data set using $\eta = 0.9$ (left) and $\eta = 0.9999$ (right).

2.6 Moving Window PCA

2.6.1 Method

MWPCA updates at each time point while restricting the observations used in the estimations to those which fall within a specified *window* of time. With each new observation, this window excludes the oldest observation and includes the observation from the previous time period. Thus, for window size H, the data matrix at time t is $\boldsymbol{X}_t = (\boldsymbol{x}_{t-H+1}, \boldsymbol{x}_{t-H+2}, \ldots, \boldsymbol{x}_t)'$, and at time $t+1$ it is $\boldsymbol{X}_{t+1} = (\boldsymbol{x}_{t-H+2}, \boldsymbol{x}_{t-H+3}, \ldots, \boldsymbol{x}_{t+1})'$. The updated $\bar{\boldsymbol{x}}_{t+1}$ and $\boldsymbol{S}_{t+1}$ can then be calculated using the observations in the new window. In a sense, the MWPCA windowing is akin to RPCA using a fixed, binary forgetting factor. While completely recalculating the parameters for each new window is straightforward, and intuitively appealing, methods have been developed to improve on computational speed (see for example Jeng (2010)). As was the case for RPCA, the model is not updated when an observation is determined to be out-of-control. A good introduction to MWPCA can be found in (Kruger and Xie, 2012, chap. 7). In particular, it includes a detailed comparison of the difference in computation time between a complete recomputation of the parameters versus an up- and down-dating approach. Both have $\mathrm{O}(p^2)$ time complexity, but in most practical situations, the adaptive approach works faster. The outlier detection and missing data methods discussed in Section 2.3 can be applied to the window of calibration data used to initialize the MWPCA model since it is assumed to be acceptably locally stationary enough to perform Static PCA modelling on.

MWPCA is characterized intuitively in Figure 2.16, where a model estimated from observations in the light-red window is used to evaluate whether the new observation, in dark-blue, deviates from typical behavior. In this characterization, at each new time point, the oldest observation is excluded from the light-red window, and the observation of the previous period is added in order to accommodate for non-stationarity. The length of the window, H, is selected based on the speed at which the mean and covariance parameters change, with large windows being well suited to slow change, and small windows being well suited for rapid change.

2.6.2 Choice of parameters

One challenge in implementing MWPCA is to select the window length H. This can be done using expert knowledge, or examination of the process by a practitioner. Chiang et al. (2001) provide a rough estimate of the window size

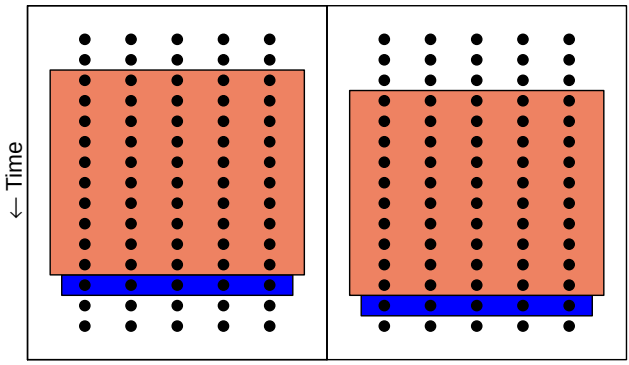

Figure 2.16: Moving Window PCA with window length $H = 10$ at times t (left) and $t + 1$ (right).

needed to correctly estimate the T^2-statistic based on the convergence of the χ^2 distribution to the F distribution that recommends minimum window sizes greater than roughly ten times the number of variables. For the Q-statistic, this window size is something of an absolute minimum, and a higher size is likely necessary. Inspired by Choi et al. (2006), He and Yang (2008) propose a variable MWPCA approach which changes the length of the window in order to adapt to the rate at which the system under monitoring changes. Once the window size is selected, the additional complication that there is not yet enough observed data may arise. One approach to address this is to simply use all of the data until the window can be filled and then proceed with MWPCA. Another, proposed in Jeng (2010), is a combination of MWPCA with RPCA such that for the early monitoring period, RPCA is used since it is not obliged to consider a specific number of observations. Then, once enough observations have been recorded to fill the MWPCA window, MWPCA is used. Jin et al. (2006) also propose an approach for combining MWPCA with a dissimilarity index based on changes in the covariance matrix, with the objective of identifying optimal update points. Importantly, they also discuss a heuristic for the inclusion of process knowledge into the control chart that is intended to reduce unnecessary updating and to prevent adaptation to anticipated disturbances.

Jin et al. (2006) elaborate on the value of reducing the number of updates in order to reduce computational requirements and reduce sensitivity to random perturbations. He and Yang (2011) propose another approach aiming to reduce the number of updates based on waiting for M samples to accumulate before updating the PCA model. This approach is intended to be used in a context

where slow ramp faults are present. In their paper, He and Yang (2011) propose a procedure for selecting the value of M.

Wang et al. (2005) propose a method for quickly updating the mean and covariance estimates for cases where the window size exceeds three times the number of variables, and of using a V-step-ahead prediction in order to prevent the model from adapting so quickly that it ignores faults when they are observed. This approach proceeds by using a model estimated at time t to predict the behavior of the system at time $t + V$ and evaluate whether a fault has occurred. The intention is to ensure that the model does not overly adapt to the data and will be able to detect errors which accumulate slowly enough to pass as normal observations at each time point. As the authors point out, using a longer window will also make the fault detection process less sensitive to slowly accumulating errors. One advantage of the V-step-ahead approach is that it can operate with a smaller data matrix than a longer window would require, so computational efficiency can be gained. However, the trade off is that the number of steps ahead must be chosen in addition to the choice of the window length.

2.6.3 MWPCA applied to the NASA data

Figure 2.17 displays the results of control charts for MWPCA models. These were fitted on the last H observations of the Phase I data (since an MWPCA model is only based on H observations), and then re-applied to the Phase I observations. As for RPCA, applying a model to observations that are not consecutive with the endpoint of the calibration period is plausible for the NASA process because the early observations are stationary. Then Phase II observations are monitored using the model. Window sizes of $H = 40$ and 80 were used to parameterize models, corresponding to one-third and two-third of the size of the calibration set. MWPCA shows slightly more stability during the Phase I monitoring when $H = 80$, reinforcing what was observed when RPCA was applied; that forgetting observations too quickly can lead to too rapidly varying models and inconsistent process monitoring. We can see that the results for the model with $H = 80$ convincingly detects the fault based on the Q-statistic at about the same time as the RPCA model with $\eta = 0.9999$ ($t = 600$), but the T^2-statistic remains more or less in control as well until about $t = 600$. Thus, the monitoring statistics of MWPCA with $H = 80$ are somewhat more consistent with each other than those of RPCA with $\eta = 0.9999$. Although the monitoring statistics become very large after $t = 600$ for the MWPCA model with $H = 40$, there tend to be more detections prior to this time point, indicating that the model is less stable than the one obtained with $H = 80$. In this respect, the results are similar to those of RPCA with $\eta = 0.9$.

Although we find in this case that MWPCA with a slower forgetting factor of $H = 80$ performs better than with $H = 40$, we also note that it has different performance than Static PCA, since it convincingly detects the fault only at around $t = 600$. This could be desireable for the reason that before $t = 600$ the vibrations in bearing four are not so great that they necessarily justify stopping the machine, but beyond this time point, the vibrations begin to increase rapidly.

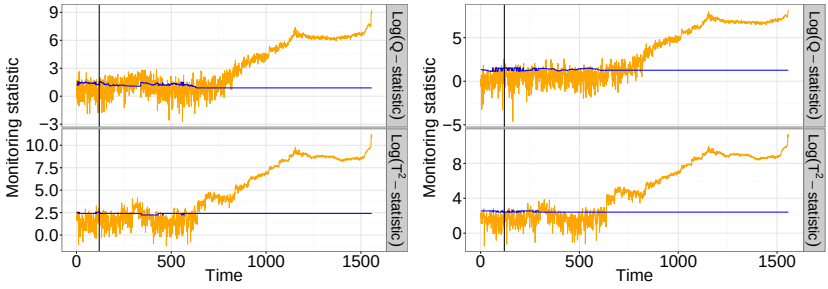

Figure 2.17: MWPCA control charts for the NASA data set using $H = 40$ (left) and $H = 80$ (right).

2.6.4 A note on the relationship between RPCA and MWPCA

Under certain conditions, we expect that a relationship exists between η and H allowing an equivalence mapping between RPCA and MWPCA. For monitoring the correlation, it can be shown that MEWMA updates of the covariance matrix can be related to moving window updates through,

$$H = \frac{2}{(1-\eta)} \tag{2.4}$$

given that the variables are uncorrelated. Under this assumption the expectation of the correlation matrices is the same, so monitoring behavior is equivalent in expectation as well. However, the extension of this equivalence to PCA is not trivial since it can be affected by non-stationarity of the process mean and covariance. This implies that an equivalent relationship would only be valid locally and thus lacking in general applicability. Nevertheless, we found that Equation (2.4) also gives a reasonable approximate mapping of RPCA to MWPCA, in terms of the resulting charts' performances. We note that a similar expression, $N = 1/(1-\eta)$, appears in the literature for defining the memory length, N, of recursive least squares and recursive partial least squares

procedures [Dayal and MacGregor (1997b); Jose (2014)]. This quantity is nothing more than 0.6321, or $1 - 1/e$, so for any forgetting parameter, N covers 0.6321 of the total weight. Though similar in appearance, this relationship is for recursively updating the mean and covariance, while the relationship in Equation (2.4) aims to match the correlation matrices of RPCA and MWPCA as much as possible to yield equivalent monitoring performance. Even though the exact relationship is not known, the proportional dependency is still very useful for interpreting the results. However, the study of the full ramifications of such equivalency goes beyond the scope of this paper.

2.7 Discussion

Among the most important questions is how to choose the optimal values of the parameters used by DPCA, RPCA and MWPCA. We have focused on illustrating the properties of these algorithms as their parameters vary by using low and high values. However, in practice an optimal value for monitoring is desired. Often, the determination of these parameters is left to the discretion of an expert on the system being monitored. Automatic methods have been described, but no consensus exists on which is the best, and further research is particularly needed in the area of automatic methods for RPCA and MWPCA parameter selection.

Methods for addressing the influence of outliers during the calibration phase exist, see e.g. Hubert et al. (2005); Jensen et al. (2007), as well as for during online monitoring (see Chiang and Colegrove (2007), Choi et al. (2006), and Li et al. (2000)). These methods address the problem of how to best make use of information captured in outliers, and approaches range from excluding them completely to down-weighting the influence exerted by such observations. Which approach is preferable, and whether different types of outliers should be treated differently are still open questions. Similarly, approaches for missing data imputation for PCA that can be applied when the calibration data is incomplete have also been proposed (Walczak and Massart (2001) and Serneels and Verdonck (2008)), but little has been done to explore the performance of these methods in the PCA process monitoring setting, or when the data is autocorrelated.

2.8 Conclusions

This chapter has covered the wide range of PCA-based approaches for monitoring and understanding high-dimensional, time-dependent processes. What has not been addressed in this chapter is the effectiveness of these methods. This deeper investigation will be carried out in Chapters 3 and 4. Specifically, Chapter 3 studies the fault detection power of important PCA-based methods, and Chapter 4 investigates how well these methods model complex process types and the sensitivities of these methods to the parametrization of those processes.

Deepening the work on parameter selection for adaptive monitoring methods is the topic of Chapter 5. There, automatic procedures for parameterization for RPCA and MWPCA are introduced, as well as an adjustment procedure for the control limits of the monitoring model. Regarding the topic of outliers in process data, later work in this dissertation addresses the specific issue of achieving this goal while performing variable selection. This topic is covered in Chapter 6, where we introduce a new method for performing spare, robust PCA, which simultaneously accounts for outliers and performs variable selection.

Chapter 3

Fault detection capabilities of PCA-based methods

Based on: Rato, T., Schmitt, E., De Ketelaere, B., Hubert, M., Reis, M. (2016). A Systematic Comparison of Statistical Process Monitoring Methods for High-dimensional, Time-dependent Processes. *AIChE Journal*, 62 (5), 1478-1493.

3.1 Introduction

A number of works in the literature provide an overview of PCA-based process monitoring or compare it to other methods [Russell et al. (2000); Kourti (2005); Ferrer (2007); Kruger and Xie (2012)] and Chapter 2, but to our knowledge none provide a broad, cross-method coverage of the behavior of even the most basic methods when applied to time-dependent processes. Given the prevalence of precisely this type of data in fields such as industry, information technology, precision agriculture, health care and economy, this chapter sets out to illustrate and compare the detection performance of fundamental methods on a collection of simple, but informative, high-dimensional, time-dependent processes.

Working with such processes allows us to provide precise insights into the drivers of detection performance. We focus on simple, fundamental methods because these are the most likely to be used in practice, and their performance is indicative of that of extensions developed to obtain, for example, greater interpretability. A broad comparison of this sort allows us to examine the relative

merits of these methods on a common basis, whereas currently one is obliged to assess them based on results from a heterogeneous collection of processes in different articles. Moreover, as some of the results obtained in this comparison study contradict expectations, revealing surprising monitoring behavior, having a clear understanding of the basic characteristics of each method is important for both practitioners and researchers developing extensions for them.

Chapter 2 introduced the main concepts of SPM and PCA-based methods, but some details that are particularly relevant to this chapter are restated and expanded upon to frame the work that follows. SPM aims to detect deviations from typical process behavior during two distinct phases of process measurement, called Phase I, and Phase II. Phase I monitoring is the practice of retrospectively evaluating whether a previously completed process was statistically in-control. On the other hand, Phase II monitoring is the practice of determining whether new observations from the process are in-control as they are obtained. During both phases, time dependence in the form of autocorrelation and/or non-stationarity can be present. Autocorrelation arises when the in-control measurements within one time series are not serially independent, while non-stationarity arises when the parameters governing a process, such as the mean or covariance, change over time. In this work, only the problem of Phase II monitoring will be addressed.

We assume that we observe a large number, p, of time series $\boldsymbol{x}_j(t_i)$, $(1 \leqslant j \leqslant p)$, typically corresponding to variables in the process, during a calibration period $t_1, t_2, \ldots, t_T$ that collectively constitute a high-dimensional data set. As time continues, more measurements become available. When the data are not time-dependent, control charts based on Principal Component Analysis (PCA) have been successfully applied in high-dimensional settings. These methods train a model on an existing data matrix $\boldsymbol{X}_{T,p}$, that is representative of typical process behavior. The j-th column contains the j-th time series $\boldsymbol{x}_j(t_i)$ for $1 \leqslant i \leqslant T$. The number of rows of $\boldsymbol{X}_{T,p}$ refers to the number of calibration observations, and p is the number of measured variables. Such methods compare a new observation at time t, $\boldsymbol{x}(t) = [\boldsymbol{x}_1(t), \boldsymbol{x}_2(t), \ldots, \boldsymbol{x}_p(t)]'$, to the data in $\boldsymbol{X}_{T,p}$, and evaluate whether it is typical. This is called static PCA because the trained model contains no dynamic components. Only the current measurement is used in the process evaluation at each time, t. Moreover, the base model remains unchanged as new observations are obtained. Therefore, no attempt is made to model relationships between observations at different time points (autocorrelation), and it will not adjust as underlying parameter values change (non-stationarity).

Three classes of approaches have been proposed to extend PCA methods to cope with time-dependent data. These are Dynamic PCA (DPCA), Recursive PCA (RPCA), and Moving Window PCA (MWPCA). DPCA was developed

to handle autocorrelation, whereas RPCA and MWPCA are intended to deal with non-stationary data. The rest of the chapter is organized as follows. In Section 3.2, performance of theoretical control limits is compared against empirical limits, a subject which is treated in more detail in Chapter 5, but which is relevant in this paper, where an *ad hoc* version of empirical limits were employed. Section 3.3 introduces an extension to DPCA based on decorrelated residuals that was also examined in this work. Section 3.4 details the step fault simulation scenarios used to compare the fault detection performance of the methods, and reports on the results obtained. These results are expanded in Section 3.5 to ramp faults. Section 3.6 illustrates the behavior of the methods on the well-known Tennessee Eastman process. In Section 3.7, we discuss the results arising from this comparison study.

3.2 Comparison of control limits from conventional and tuned alpha values

Chapter 2.3 introduced basic PCA notation and the classical derivations for the control limits. However, when monitoring complex processes, a perfect model is rarely achieved and the assumptions of the classical limits can be violated. One way to address this is to use empirical limits. Throughout our simulations, we select values of α_{T^2} and α_Q so that FDR_{T^2}=FDR_Q=0.005 and the global FDR=0.01. Conventionally, one would expect to achieve an FDR of 1% by simply setting $\alpha_{T^2} = \alpha_Q = 0.005$. We will illustrate in the following example that this conventional approach leads to an unacceptably high FDR on non-stationary data. We fit RPCA and MWPCA models to an IMA(1,1) process using the forgetting factors given in Table 3.3, and implement two monitoring schemes with different control limits. In Table 3.1, we summarize the performance of the adaptive methods using conventional and tuned limits. Conventional limits lead to undesirably high detection rates, but the tuned limits give close to 1% false detection. When we examine the α values selected by our algorithm, we see that in this case those of for the limits for the Q-statistics are significantly smaller than convention would dictate. Figures 3.1 and 3.2 present the control charts corresponding to these results. We notice that the control charts with tuned limits have consistent monitoring statistics, while the Q-statistic charts based on conventional limits detect too many faults.

Table 3.1: False discovery rate of PCA control charts on the ARI process.

Method	Forgetting factor	Limits	α_{T^2}	α_Q	DR_{T^2}	DR_Q	DR_g
RPCA	0.999	conventional	0.005	0.005	0.7%	8.4%	9%
RPCA	0.999	tuned	0.004	1.67×10^{-6}	0.8%	0%	0.8%
MWPCA	800	conventional	0.005	0.005	0.6%	7.5%	8.1%
MWPCA	800	tuned	0.002	9.62×10^{-7}	0.2%	0%	0.2%

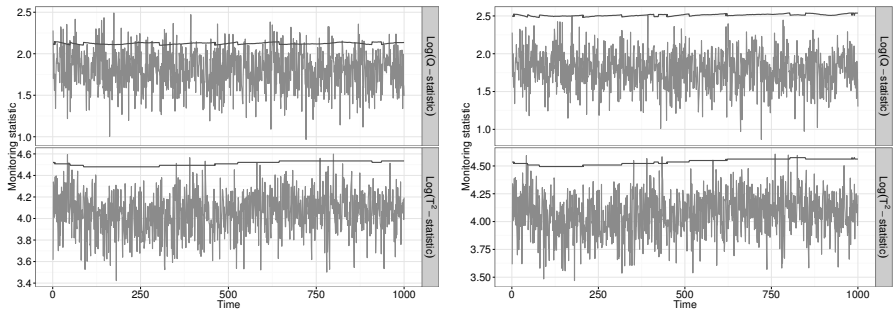

Figure 3.1: RPCA control charts based on conventional (left) and tuned (right) α values applied to an IMA(1,1) process.

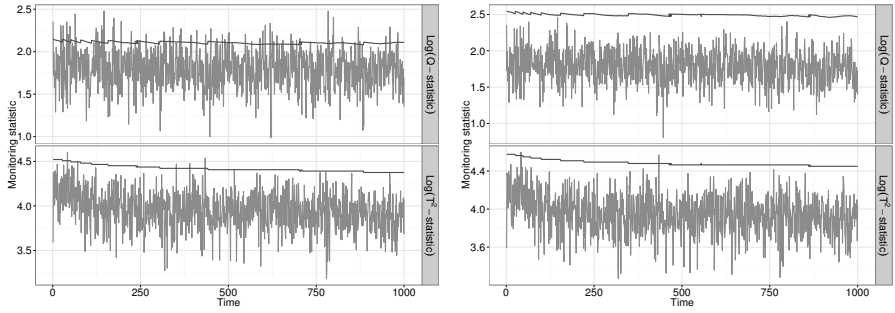

Figure 3.2: MWPCA control charts based on conventional (left) and tuned (right) α values applied to an IMA(1,1) process.

3.3 Dynamic PCA with decorrelated residuals

Chapter 2 introduced static PCA and some of its extensions, including DPCA. Here, an extension to DPCA that is also treated in this comparison is introduced.

The reason for including this approach as well is that introducing lagged

variables in DPCA allows the description of the autocorrelation present in the data. However, the T^2 and Q statistics can still exhibit autocorrelation. In the case where enough lags are selected, the Q-statistic should indeed have no serial correlation. Yet, even in this case, there is autocorrelation in the scores, and subsequently in the T^2-statistic. To overcome this issue, Rato and Reis (2013c) proposed a combination of DPCA with a missing data estimation technique [Nelson et al. (1996); Arteaga and Ferrer (2002)] in order to obtain better time-decorrelated statistics that they call DPCA-DR. In this method, a DPCA model is constructed as in the previous section, and from it we obtain the usual scores $\boldsymbol{y}_k = \boldsymbol{P}_k'(\boldsymbol{x} - \bar{\boldsymbol{x}})$. An additional vector of estimated scores $\hat{\boldsymbol{y}}_k$ is computed by assuming that the current observation vector $\boldsymbol{x}(t)$ is missing. This is a one-step-ahead prediction of the scores based on the implicit AR model estimated by DPCA. Moreover, the application of this methodology gives an estimate of the scores that best agree with the last l known measurements. Given these scores, the following Hotelling's T^2 statistic is defined:

$$T^2_{prev} = (\boldsymbol{y}_k - \hat{\boldsymbol{y}}_k)' \boldsymbol{S}^{-1}_{prev} (\boldsymbol{y}_k - \hat{\boldsymbol{y}}_k),$$

where $\boldsymbol{S}_{prev}$ is the sample covariance matrix of the difference between the observed and estimated scores, $(\boldsymbol{y}_k - \hat{\boldsymbol{y}}_k)$, that monitors the DPCA reference subspace. Control limits for this statistic can be determined empirically to obtain the desired false detection properties. Likewise, a monitoring statistic for the residual subspace, which replaces the Q-statistic, is defined as:

$$T^2_{res} = \boldsymbol{r}' \boldsymbol{S}^{-1}_{res} \boldsymbol{r} = (\boldsymbol{x} - \boldsymbol{P}_k \hat{\boldsymbol{y}}_k)' \boldsymbol{S}^{-1}_{res} (\boldsymbol{x} - \boldsymbol{P}_k \hat{\boldsymbol{y}}_k) \tag{3.1}$$

where $\boldsymbol{S}_{res}$ is the sample covariance matrix of the residuals in the reconstructed data, obtained with the estimated scores $(\boldsymbol{r} = \boldsymbol{x} - \boldsymbol{P}_k \hat{\boldsymbol{y}}_k)$. Limits for this statistic are also determined empirically. Xie et al. (2006) introduce another PCA-based monitoring approach based on subspace identification that achieves decorrelation results similar to DPCA-DR. It is a more complex modelling approach but has the advantage of producing a final model that allows for greater interpretability.

3.4 Simulation studies

In this section, we evaluate the performance of the PCA-based methods on a variety of pure time-dependent processes. This was intentionally done because it allows for complete control of the data generation, and eliminates confounding behavior that can arise in simulations that attempt to model complex real world processes. We find that even though we are restricting ourselves to this limited set of scenarios, the study yields useful and surprising insights. The process

settings and fault scenarios are varied to provide a comprehensive overview of how they affect performance for each process scenario. The intention is to provide an overview of potential situations the practitioner might encounter, and to illustrate the type of monitoring performance that can reasonably be expected from the methods we consider.

We simulate faults where they appear in complex systems: in the inner latent components (the scores, $\boldsymbol{y}_t$) and measurement sensors (the data, $\boldsymbol{x}_t$). Sensor faults are self-explanatory. Conceptually, the scores represent latent structures of the process (fundamental structures in the process that cannot be directly observed). Thus, while sensor faults may indicate an issue with a particular sensor or element of the process, score faults indicate systematic faults. Five types of time-dependent processes are considered: an autoregressive (AR) process, a moving average (MA) process, an autoregressive process with a unit root (ARI), an integrated moving average (IMA) process, and a process that is non-stationary in the loadings structure (NSS). Following convention (e.g. Burnham et al. (1999); Choi et al. (2006)) we generate data at the subspace level so that we can explicitly control the features monitored by the PCA models, though we do not make use of this knowledge when monitoring. Doing so makes it possible to compute objective performance metrics, as we know the true underlying behavior of the system under Normal Operating Conditions (NOC) and faulty conditions. To obtain each observation at time t we began by generating five scores, $\boldsymbol{y}_t$, according to the equation of the desired process. For all process types, we introduce variation onto the process dynamics through $\boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}_5, 0.01\boldsymbol{I}_5)$, where $\boldsymbol{I}_5$ is the 5×5 identity matrix. These are then transformed into a 100-dimensional data set of measurements computed as

$$\boldsymbol{x}_t = \boldsymbol{P}_0 \boldsymbol{y}_t + \boldsymbol{e}_t, \tag{3.2}$$

where $\boldsymbol{P}_0$ is a 100×5 matrix with orthogonal columns randomly generated once and kept constant for all simulation runs over all processes. The decision to simulate 100-dimensional data was motivated, in part, by a lack of other studies considering high-dimensional processes, even though PCA is often proposed for precisely that scenario. The $\boldsymbol{e}_t$ are 100×1 vectors of white noise errors, distributed as $\mathcal{N}(\mathbf{0}_{100}, 0.000025\boldsymbol{I}_{100})$, that simulate measurement noise, as is done, for instance, in (Ku et al. (1995) and Lakshminarayanan et al. (1997)). The $\boldsymbol{e}_t$ can be seen as the error at the sensor level, and are set to a small value here under the assumption that sensors are typically reliable. For all methods and simulations, a CPV of 95% is used. Since the statistics are not guaranteed to be i.i.d. in the dynamic context, an alternative to the analytical expressions for the limits and the chosen α level is necessary. The control limits of the non-adaptive methods are determined based on a validation data set with 5000 NOC observations. A different approach was necessary to ensure that the adaptive methods also achieved the desired False Detection Rate (FDR)

on NOC data since the control limits are not constant. To do so, we search for values of α for the T^2 and Q (call these α_{T^2} and α_Q), which, when used as input in the analytical expressions for the control limits of these statistics, result in an FDR_{T^2} and an FDR_Q whose sum equals the desired total FDR for the model. To accomplish this, we impose that FDR_{T^2}=FDR_Q, and assume that T^2 and Q are independent, which is plausible for these simulations. Note that the adjusted values of α_{T^2} and α_Q do not correspond to the statistical significance of the monitoring statistics, since the theoretical expressions for the control limits are not valid under the conditions simulated in this study. Further, they do not need to be equal to each other. These limits are set such that the false detection rate of each method was 1% (i.e., the combined use of scores and residual statistics through a logical gate OR gives an overall false detection of about 1%).

The number of lags for the dynamic PCA methods was selected using the method of Rato and Reis (2013b). To select the values of η and H, we considered a range of possible values and their corresponding α values, and selected one giving good monitoring results, in terms of false detection rate, on a validation NOC data set. Although the selection of forgetting parameters for adaptive methods is critical to their proper implementation, this topic is not well covered in the literature.

Faults are introduced to the process on either the first score in $\boldsymbol{y}_t$, or the first measurement variable (sensors) in $\boldsymbol{x}_t$, by simple addition of a step deviation with magnitude defined as d times the standard deviation of the first element of $\boldsymbol{\varepsilon}_t$ for score faults, and $\boldsymbol{e}_t$ for sensor faults.

This approach of varying d gives a sense of how difficult it is to detect a fault for a given process type (how large the fault would needs to be), and provides a basis for comparison between methods and across process types. We expect that the T^2 and T^2_{prev} statistics will detect the score faults, and the Q and T^2_{res} statistics will detect the measurement faults, because they monitor distance of the observations on the model subspace and from it, respectively. Each process type, fault type, and value of d constitutes a scenario. Each faulty data set contains 1000 observations (the first 500 under NOC and the remaining 500 under the effect of a fault). One hundred faulty data sets are generated for each scenario to assess the stability of the results. We considered increasing values of d to illustrate how the methods behave as the faults become more clear. Negative deviations were also investigated, and yielded symmetrical results. An overview of the parametrization used for each of the methods on the simulation scenarios we consider are given in Tables 3.2 and 3.3.

In the above tables, LV stands for the number of latent variables. Parameter values are left blank in the ARI(1,1) and IMA(1,1) cases for the PCA and

Table 3.2: Parameter settings for non-adaptive methods.

	PCA	DPCA		DPCA-DR	
Process	LV	LV	Lags	LV	Lags
AR(1)	5	9	0–2	9	0–2
MA(1)	5	46	10	46	10
ARI(1,1)	- -	- -	- -	4	50
IMA(1,1)	- -	- -	- -	37	10
NSS	14	144	10	144	10

Table 3.3: Parameter settings for adaptive methods.

	RPCA		MWPCA	
Process	LV	η	LV	H
AR(1)	5	0.9987	5	700
MA(1)	5	0.9999	5	800
ARI(1,1)	4-5 (4)	0.995	4-5 (4)	530
IMA(1,1)	5	0.999	5	800
NSS	7-14 (11)	0.93	5-12 (7)	102

DPCA models because in practice they were found to be unsuitable for these processes and no simulation results are presented. In the case of RPCA and MWPCA ranges and averages are given for the ARI and IMA cases, where the number of components changes as the process evolves.

We are particularly interested in the detection rates (DR: not to be confused with the decorrelated residuals of DPCA-DR) of the methods. In order to compare them visually, we will plot the average DR (i.e., the ratio between the number of alarms over the number of faulty observations) of each of the methods as a function of the deviation size for the 500 time points after the fault is introduced. Additionally, whiskers are plotted around the lines indicating the 25% and 75% quantiles. These detection rates correspond to the True Positive Rate (TPR), and one minus these detection rates gives the False Negative Rate (FNR) (the rate at which faults are incorrectly classified as in-control). Similarly, the False Positive Rate (FPR) can be inferred from the zero deviation case. To assess the monitoring statistics behavior over time, we also present plots with the percentage of runs that correctly give an out-of-control signal at each faulty time period for the largest magnitude deviations applied to the scores and measurements of each process we studied. The x-axis is time on a log scale to highlight the detection power of the methods at the earliest time points, where it is most relevant. Therefore, these plots are meant to give a sense of how quickly on average each of the methods detects the faults. Monitoring results prior to the introduction of the fault are not shown since these conform

on average to the false detection rate of 1% that we selected.

3.4.1 AR(1) process

The AR process is investigated because of its natural relevance for studying the properties of DPCA and DPCA-DR. Furthermore, this is a particularly relevant process type because the high sampling rate of many contemporary sensors inherently introduces autocorrelation into the data. The AR(1) process is defined as [Box et al. (1994)]:

$$\boldsymbol{y}_t = \phi \boldsymbol{y}_{t-1} + \boldsymbol{\epsilon}_t, \tag{3.3}$$

where $\boldsymbol{y}_t$ are the serial observations of the underlying latent model ($\boldsymbol{y}_t$ in Equation (3.2)) and ϕ is the AR coefficient. The NOC score and sensor behavior of this process is depicted in Figure 3.3.

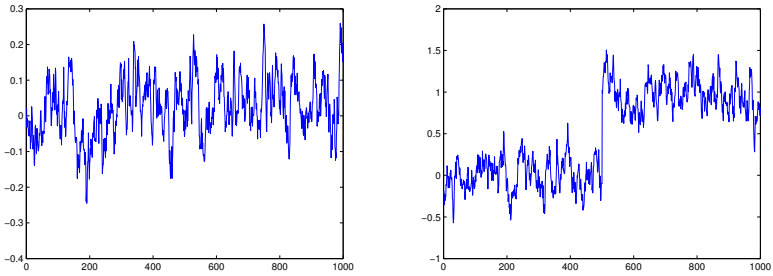

Figure 3.3: The first variable (left) and the first score (right) for an AR(1) process.

In simulating the AR(1) process, five scores with a relatively high AR coefficient ϕ equal to 0.90 were used to generate the data according to Equation (3.3). Although negative autocorrelation is a possibility, we do not consider it since the rapid sampling rate of modern processes means that positive autocorrelation is far more common.

Figure 3.4 displays the obtained fault detection rates for the 100 replications. It shows that static PCA and DPCA are both capable of detecting the simulated score faults at approximately the same level of accuracy. The Q-statistic corresponding to static PCA does not exhibit autocorrelation and for DPCA only small levels of autocorrelation are observable. This situation means that both models are successful in describing the system. Still, the Hotelling's T^2 is highly autocorrelated, which undermines the detection of deviations at the

scores level, leading to weak detection of faults on the score subspace. As a consequence, large values of d are necessary before good detection results are observed. On the other hand, DPCA-DR produces monitoring statistics without significant autocorrelation but is only able to detect faults much larger than the ones presented here (for instance, a fault of magnitude 40 standard deviations has, on average, a detection rate of 0.63). This is a direct result of the DPCA-DR estimation step and subsequent differencing between the observed and estimated scores (see Equation (3.1)). Thus, when a fault is introduced, the estimated scores by missing data do not comply with the full data scores, causing the T^2_{prev}-statistic to signal an alarm. However, after this initial detection, the subsequent scores fitted by DPCA-DR begin to resemble faults, since the previous, faulty observation is added as a lag and ultimately T^2_{prev} returns to in-control status, as seen in the left plot of Figure 3.5, which illustrates the rate of fault detection for the time period after the fault is introduced. A similar, smaller, adaptation is observed for DPCA as well, but it still produces a detection rate on par with PCA. On the other hand, DPCA-DR is far more capable of detecting sensor faults than other methods, while PCA and DPCA do not show a large increase in detection after the fault is introduced for the deviation values displayed in the plot (they can fully detect larger faults not covered by that range).

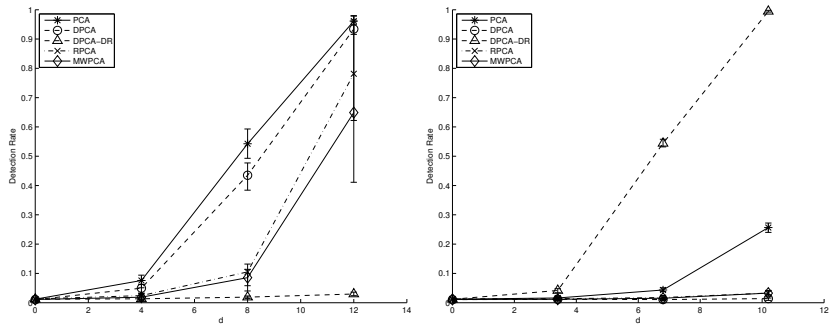

Figure 3.4: Fault detection rate curves for step deviations on one of the scores (left) and sensor measurements (right) of the AR(1) process. The fault magnitude is defined as d times the standard deviation.

There is an observable difference between the behavior of RPCA and MWPCA and the non-adaptive methods. Considering the left plot of Figure 3.4, we see that the adaptive methods do not detect score faults as accurately as PCA or DPCA, though between RPCA and MWPCA there is no notable difference for this type of fault. The right plot of Figure 3.4 shows RPCA and MWPCA producing equally weak sensor fault detection results. Consulting both plots in Figure 3.5, it is clear that the reason for the weaker detection of score faults

is because RPCA and MWPCA adapt to the fault after initially detecting it. This may not be an issue since detection in the moments after a fault occurs is of primary interest. The adaptation of the methods to the faults occurs because low values of the faulty observations are still within the old control limits, but these are relatively high in the context of the distribution of the NOC period statistics and therefore pull the control limits higher, allowing yet more faulty observations to enter the updating calculations.

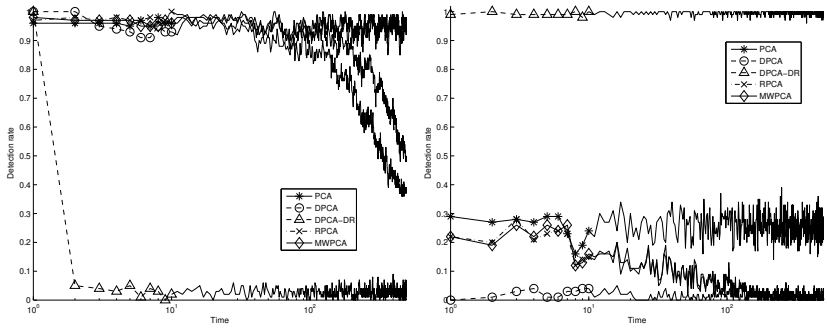

Figure 3.5: Score (left) and sensor (right) fault detection rate on the maximum simulated score and sensor faults for the AR(1) process at each faulty time period averaged over all runs.

Interestingly, these results illustrate the robustness of static PCA in detecting faults in systems with simple dynamics, even when the monitoring statistics exhibit autocorrelation. One would expect DPCA to outperform it by a considerable margin, but in practice the difference is small. Therefore, PCA is regarded as the most suitable monitoring scheme for detecting score faults in this case scenario. This is in fact in line with Russell et al. (2000), who provided a comparison of DPCA against PCA on simulated data from the Tennessee Eastman process in which they find that the two methods give similar results. DPCA-DR offers the best detection of sensor faults.

3.4.2 MA(1) process

Like AR processes, MA processes are fundamental time-dependent processes. The MA(1) process is defined as [Box et al. (1994)]:

$$\boldsymbol{y}_t = \boldsymbol{\varepsilon}_t - \varphi \boldsymbol{\varepsilon}_{t-1}. \tag{3.4}$$

where φ is the MA coefficient. The NOC score and sensor behavior of this process is depicted in Figure 3.6.

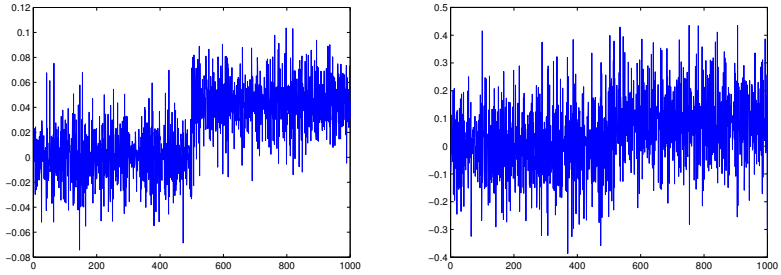

Figure 3.6: The first variable (left) and the first score (right) for an MA(1) process.

If an MA process is invertible, it can be equivalently described as an AR process with an infinite order. Moreover, as at some point the equivalent AR coefficients may become close to zero, DPCA methods should be able to model them under these conditions. The φ in Equation (3.4) is set to 0.90 for all of the scores. Note that none of the procedures covered in this study are specifically designed for modeling MA processes.

Figure 3.7 reveals that PCA is incapable of detecting score faults. This may come as a surprise since this MA(1) can be reformulated as an AR process with weaker autocorrelation than 0.9, and we have observed in the AR(1) case that PCA is capable of providing good monitoring. However, to monitor the MA(1) process, far more lags may be necessary even though the autocorrelation is weaker, and since PCA has no lags, it cannot model the process well. Indeed, we find that when applying DPCA, more lags are needed to model the MA(1) process than the AR(1), highlighting their importance in modeling this process. The score fault detection power of DPCA is the highest of the methods in this scenario. However, autocorrelation on the monitoring statistics is still present. The performance of PCA and DPCA in the MA(1) case is related with their relative capability to properly model the process. For faults in the scores, no observable change occurs in their T^2-statistics, which indicates that both models are unable to extract the correct scores structure. Therefore, faults introduced at the scores level are not explained by the DPCA model, but ultimately translated into deviations on its Q-statistic. For the sensor faults, PCA is again more accurate than DPCA. DPCA-DR displays the strongest sensor faults, and decent detection for score faults. This greater performance in score detection happens because the DPCA-DR model has the same deficiency as DPCA in what regards their computation, i.e., the scores remain consistent with the behavior of the system before the fault. However, the missing data

estimation stage intrinsic to the method is able to follow the fault, due to faults in the lags, leading to different estimates of scores. As a result, an out-of-control state is observed since the difference between these two values is assessed and captured by both the T^2_{prev} and T^2_{res} statistics. We note that this is a different result from the AR(1) case, where both approaches of DPCA-DR to compute the scores eventually became consistent and so fault detection diminishes. Figure 3.8 shows that both DPCA and DPCA-DR exhibit some delay before they completely detect the introduced score faults, and we see a similar delay for DPCA in the case of the sensor faults. This is because at each time period a new lag with a fault in it is included. The initial time periods following the fault still have a significant number of normal observations in the vector of lagged observations. However, once enough lags contain faulty observations, the scores become sufficiently distant from the estimated subspace that they can be detected using the residual statistics.

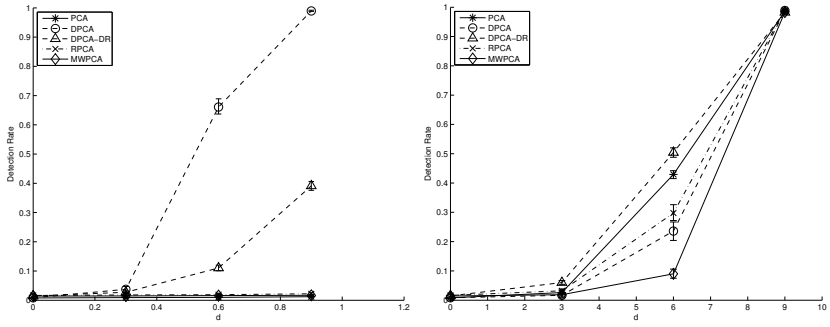

Figure 3.7: Fault detection rate curves for step deviations on one of the scores (left) and sensor measurements (right) of the MA(1) process. The fault magnitude is defined as d times the standard deviation.

Just like PCA, the adaptive methods are not capable of determining the scores correctly and are therefore unable to detect faults in them. Both show roughly the same ability to detect sensor faults, having decent detection rates for this scenario, with a slight advantage for RPCA. As the right plot of Figure 3.8 shows, RPCA and MWPCA with the selected forgetting factor and window size do not adapt to these simulated faults when the magnitude of the fault is high. However, we observed that different parameters lead to substantially different results, which highlights the need for their proper selection.

Considering these results, for this process we recommend DPCA for monitoring for score faults, and PCA or DPCA-DR for monitoring for sensor faults. A general comment on the monitoring results for the MA(1) process is that smaller deviations can be detected than in the AR(1) case. This is because the process

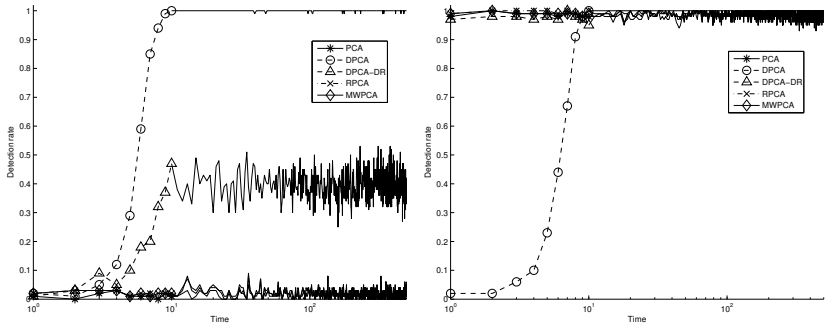

Figure 3.8: Score (left) and sensor (right) fault detection rate on the maximum simulated score and sensor faults for the MA(1) process at each faulty time period averaged over all runs.

variation is lower, making faults in the MA(1) relatively more obvious than in the AR(1) case. We shall see that subsequent processes with different process variation obey this relationship between process variation and ease of fault detection as well.

3.4.3 ARI(1,1) process

DPCA and DPCA-DR are designed for monitoring autocorrelated data. A unit root is added to the autocorrelated series to evaluate how well they perform when non-stationarity is present, and conversely to explore the performance of RPCA and MWPCA when non-stationary data is autocorrelated. The ARI(1,1) process is defined as [Box et al. (1994)]:

$$\boldsymbol{y}_t = \boldsymbol{y}_{t-1} + \phi(\boldsymbol{y}_{t-1} - \boldsymbol{y}_{t-2}) + \boldsymbol{\varepsilon}_t. \tag{3.5}$$

The latent variables of an ARI(1,1) process are generated with ϕ equal to 0.90 for all scores. Due to the time-dependent characteristics of this system, parameter determination during the modeling stage, such as selection of the number of latent variables and lags, heavily influences the performance of the final monitoring statistics. The NOC score and sensor behavior of this process is depicted in Figure 3.9.

Even when these parameters are optimally selected, most of the monitoring statistics follow a non-stationary pattern; namely the statistics of PCA and DPCA. This makes PCA and DPCA extremely unreliable since they produce a large number of false alarms when the system is under NOC. Moreover, there is no visible change in these monitoring statistics when a fault occurs. Therefore,

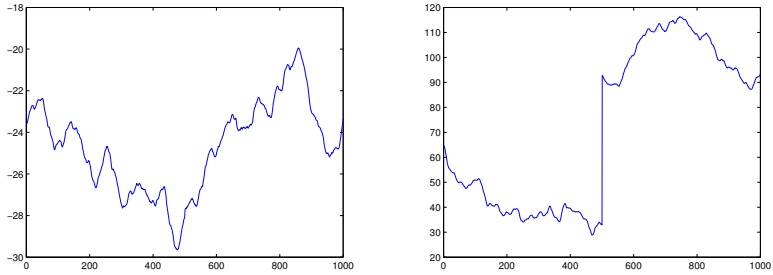

Figure 3.9: The first variable (left) and the first score (right) for an ARI(1,1) process.

PCA and DPCA models, with a single constant control limit are not suitable for this process. Consequently, PCA and DPCA are excluded from the main comparison. Since the process variation of the ARI(1,1) process is so high relative to the variation of the error terms, it happens that a CPV of 95% does not always include all five important components. Thus DPCA-DR, RPCA and MWPCA are all observed to take between four and five components.

As shown in Figure 3.10, the estimated DPCA-DR model tends to perform better under these conditions. Even though DPCA-DR was not specifically designed for ARI processes, by simple manipulation of Equation (3.5) an AR like structure can be obtained. More precisely, current observations can be defined as a linear combination of their lagged versions. This is the underlying assumption of the DPCA model used by the DPCA-DR, which fits appropriate loadings to explain the data. However, due to the higher process complexity, a significantly large number of lags is required to accurately describe it. The monitoring statistics have low autocorrelation and are generally beneath their respective control limits for NOC. Figure 3.11 reveals that DPCA-DR does initially detect score faults, however when all lags contain faulty observations, the detection capacity of this method becomes very low. It is also observed that the monitoring performance is highly irregular and dependent of the amount of faulty observations in the lagged variables. When the process is subjected to step deviations on the sensor measurements, DPCA-DR is the only method effectively capable of detecting them.

The left plot of Figure 3.10 shows that, as expected for a non-stationary process, RPCA and MWPCA both detect score faults quite well, and the average behavior of the methods is essentially the same. We see though, that both methods do display high variability, meaning that on some runs the methods do much better or worse than usual at detecting the fault. This result is a

symptom of the non-stationarity of the data: the fault may occur during either periods of high or low volatility, and may go with or against the direction of the non-stationarity. In the event that the fault moves counter to the direction of the process non-stationarity it may be masked by the movement of the process, but if the fault moves with the non-stationarity it may stand out more because it is amplified by the process. This effect is most visible for $d = 400$, which lies between smaller fault magnitudes that are rarely detected and $d = 600$, which is nearly always detected. Two reasons explain the high magnitudes of the d values. First, process variation dominates the variation of error terms, which are based on a reference i.i.d. distribution. Secondly, the even though RPCA and MWPCA are adaptive, they still cannot completely account for the non-stationarity in the data. As a consequence, the control limits are set high enough to attain the desired FDR on NOC data, but at a cost to detection power. In the right plot, we see that the adaptive methods do not display the same aptitude for detecting sensor faults, especially when compared to DPCA-DR. Figure 3.11 shows that the adaptive methods slowly adapt to score faults on some runs. In the case of sensor faults, only some of the simulation runs result in the process going out of control since the d value for which results are displayed show a transition point between d values where almost no detection occurs and full detection occurs. The runs for which faults were detected were determined to be completely out of control after the introduction of the fault, while the remaining runs remained in control.

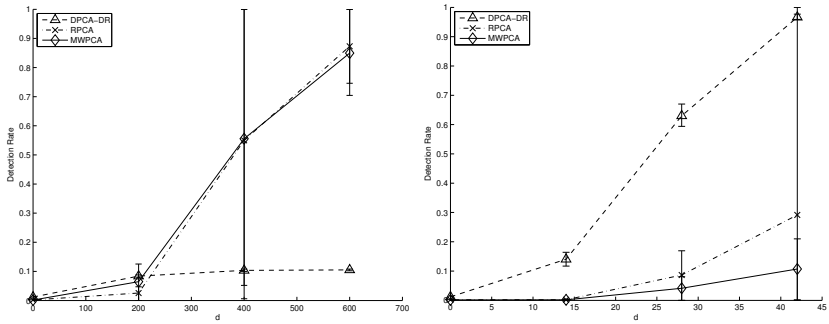

Figure 3.10: Fault detection rate curves for step deviations on one of the scores (left) and sensor measurements (right) of the ARI(1,1) process. The fault magnitude is defined as d times the standard deviation.

Based on the results, we recommend RPCA and MWPCA for detecting ARI(1,1) score faults and DCPA-DR for detecting measurement deviations.

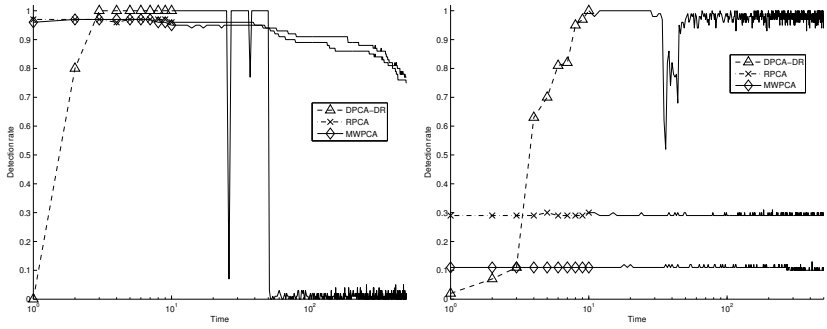

Figure 3.11: Score (left) and sensor (right) fault detection rate on the maximum simulated score and sensor faults for the ARI(1,1) process at each faulty time period averaged over all runs.

3.4.4 IMA(1,1) process

Next, a multivariate IMA(1,1) process is considered. In the univariate context, this is the process type for which the popular EWMA gives the optimal one-step-ahead prediction. This process is defined as [Box et al. (1994)]:

$$\boldsymbol{y}_t = \boldsymbol{y}_{t-1} + \boldsymbol{\varepsilon}_t - \varphi\boldsymbol{\varepsilon}_{t-1}. \tag{3.6}$$

As in the discussion of the ARI(1,1) process, the non-stationarity of the process has a greater impact on the final monitoring statistics behavior than the autocorrelation. This is particularly noticeable in the T^2-statistics, which exhibit different mean values across replications, because the process means of the simulated series differ from each other due to the non-stationarity. Therefore, as for the ARI(1,1) process, it is not possible to determine a reliable control limit for T^2_{PCA} and T^2_{DPCA} based on historical data with a different mean and variance level, even though the process structure remains the same. Thus, these methods are not considered for this process type. In Figure 3.12, we see that DPCA-DR can detect sensor faults, with the advantage of both T^2_{prev} and T^2_{res} being serially decorrelated and capable of coping with the process dynamics. This result is achieved because the IMA(1,1) is interpreted in the DPCA step of DPCA-DR as an MA(1) process in much the same way as for the ARI(1,1) process. This leads to correct estimation of the latent variables even though the base model remains unchanged throughout the simulation.

RPCA and MWPCA detect score faults well. Under our parametrization, MWPCA detects for score faults somewhat better than RPCA. Neither method is competitive with DPCA-DR for sensor faults. High variability in score

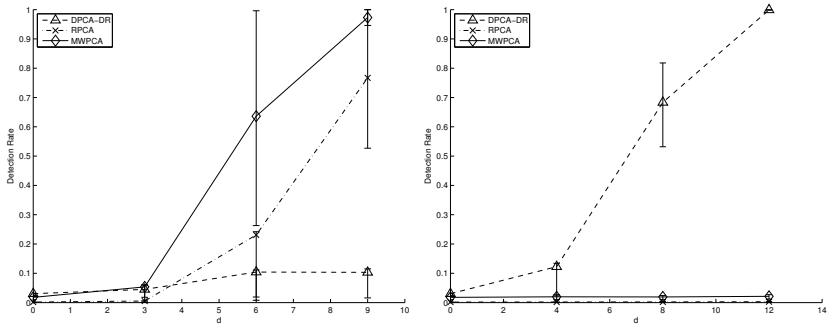

Figure 3.12: Fault detection curves for step deviations on one of the scores (left) and sensor measurements (right) of the IMA(1,1) process. The fault magnitude is defined as d times the standard deviation.

fault detection performance, similar to the detection performance noted for the ARI(1,1) process, is observed. This is again due to the non-stationarity of the process.

The results in Figure 3.13 match what we observe with the fault detection curves; namely that the adaptive methods perform well on the score faults and DPCA-DR does well on the sensor fault. Given the above results, we recommend RPCA or MWPCA for monitoring the scores and DPCA-DR for the measurements.

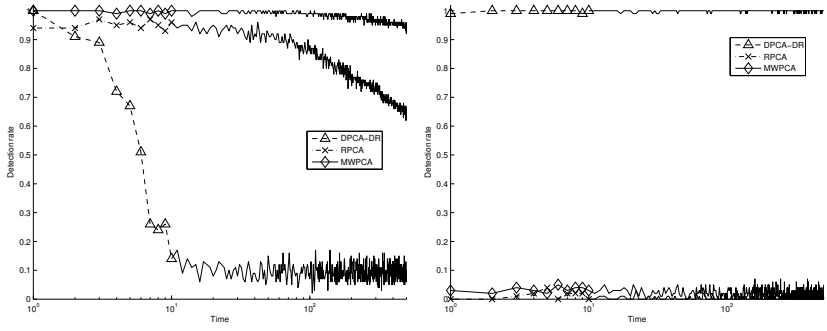

Figure 3.13: Score (left) and sensor (right) fault detection rate on the maximum simulated score and sensor faults for the IMA(1,1) process at each faulty time period averaged over all runs.

3.4.5 NSS process

Next, we consider a process that is non-stationary in the loadings structure (NSS) as opposed to the ARI and IMA processes, which introduce non-stationarity at the score level. The performance of the methods is also studied when another type of non-stationarity is present to see if they still perform as they did in the previous two scenarios. The process we consider is non-stationary locally, but exhibits a periodic fluctuation that can be considered stationary on the larger scale. The NSS process expressing the described behavior introduces non-stationarity in the form of rotations on the base latent variables hyperplane, $\boldsymbol{P}_0$ (see Equation (3.2)). By application of such rotations, the latent variables hyperplane experiences a periodic fluctuation over all its axes. These rotations are achieved by premultiplication of $\boldsymbol{P}_0$ over time by a rotation matrix, $\boldsymbol{R}(\boldsymbol{\theta}) = \boldsymbol{R}_1(\theta_1) \cdot \boldsymbol{R}_2(\theta_2) \cdot \ldots \cdot \boldsymbol{R}_{p-1}(\theta_{p-1})$, defined by the vector of angles $\boldsymbol{\theta} = [\theta_1, \theta_2, \cdots, \theta_{p-1}]'$, where,

$$\boldsymbol{R}_1(\theta_1) = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 & \cdots & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix}, \quad \boldsymbol{R}_2(\theta_2) = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & \cos\theta_2 & -\sin\theta_2 & \cdots & 0 \\ 0 & \sin\theta_2 & \cos\theta_2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix}, \text{etc}$$

For this case study, θ_i was generated as:

$$\theta_i = A_i \sin(2\pi f_i t)$$

where A_i is the amplitude, t is the sampling time and f_i is the frequency.

In this case we set the amplitude to $15\pi/180$, which corresponds to a $\pm 15\circ$ rotation on the base hyperplane, and the frequency to $1/1000$ (i.e., a full rotation is obtained at every 1000 observations). Since we are using 5000 observations to train the models, this corresponds to five complete periods. Each score was generated from a normal distribution with zero mean and variance 0.01 in order to make it comparable with the other processes. The NOC score and sensor behavior of this process is depicted in Figure 3.14.

As for the previous cases study, faults were introduced in one of the scores or one of the measurements. The detection rates for these faults are depicted in Figure 3.15.

Since PCA and DPCA-DR are based on static models, they can only accurately describe the process over some local regions resembling the average behavior of the process even though the process exhibits periodic rotation. Therefore, whenever the fluctuation causes the data to depart from this average behavior,

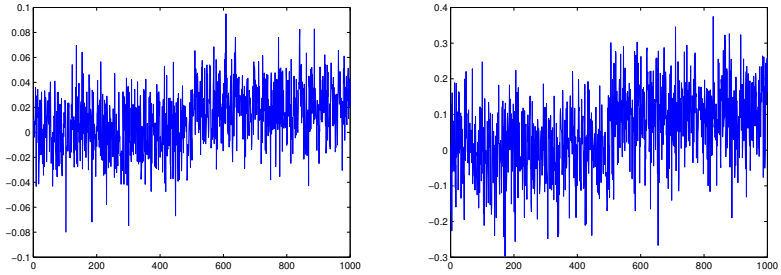

Figure 3.14: The first variable (left) and the first score (right) for an NSS process.

the monitoring statistics increase. This happens in a cyclic way and is visible in the left plot of Figure 3.16, and to a lesser extent in the right plot. This situation also causes a generally lower detection rate for PCA and DPCA-DR because the control limits are inflated and thus mask the faults when they occur in periods closer to the reference model. Still, DPCA-DR shows a competitive performance, which resembles PCA in score faults and DPCA in sensor faults. As for DPCA, the periodic effect is mitigated, allowing it to have a high performance in both type of fault.

Since they are adaptive methods, the RPCA and MWPCA models should be able to adjust as the base hyperplane rotates. In order to achieve this performance, a forgetting factor or window size related to the rotation frequency should be used. For our NSS process, this requires the use of small parameter values. Figure 3.15 shows that these methods have weak fault detection as a consequence of their high adaptation, and therefore they are only detecting faults with large deviations. As shown in the right plot of Figure 3.16, both weakly detect the introduced score faults at their inception, but later adapt to them. The right plot of Figure 3.16 shows the adaptive methods obtaining a low detection rate.

Based on the results, we recommend the use of DPCA for detecting both types of faults.

3.5 Simulations with ramp faults

In this section we redo our earlier simulations, but introduce ramp faults instead of step faults. These results are similar to the step fault results,

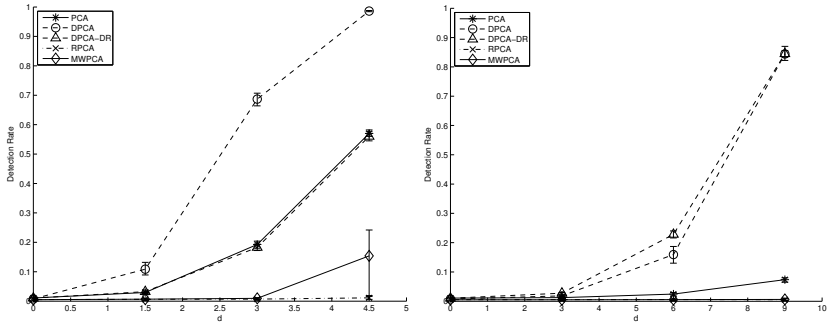

Figure 3.15: Fault detection rate curves for step deviations on one of the scores (left) and sensor measurements (right) of the NSS process. The fault magnitude is defined as d times the standard deviation.

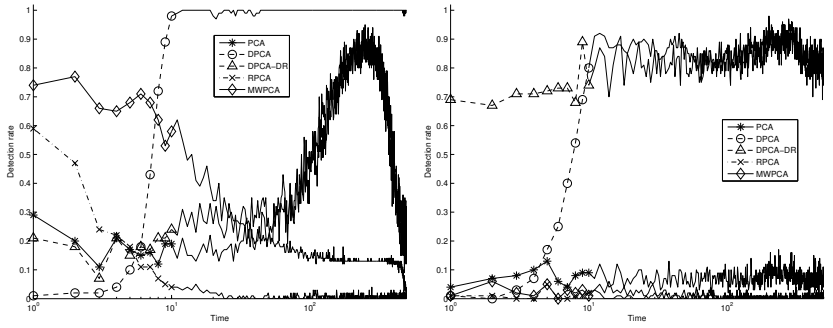

Figure 3.16: Score (left) and sensor (right) fault detection rate on the maximum simulated score and sensor faults for the NSS process at each faulty time period averaged over all runs.

with the main difference being that RPCA and MWPCA have weaker fault detection performance because they adapt to the ramp faults as they occur. The performance of the non-adaptive methods on ramp faults and step faults differs in that larger magnitude faults are needed to detect the ramp faults earlier. Ramp faults are generated with a slope such that they take 500 observations to reach the full fault magnitude. Therefore, the larger the final magnitude, the higher the slope of the fault, and the more it resembles a step fault.

3.5.1 AR

In Figures 3.17 and 3.18 we show the detection rates and speed of detection for ramp faults on the AR(1) process. The ranking of the methods in terms of score and sensor fault detection is the same as in the step fault scenario. However, we note that the faults must now have much higher magnitudes to be detected during the entire fault period. This is because at the beginning of the fault the deviation is not yet fully developed.

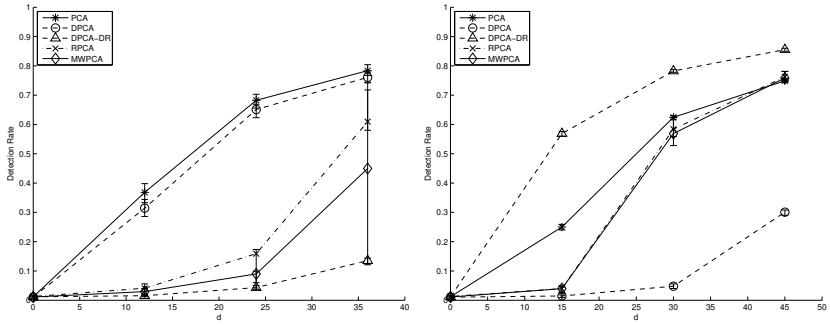

Figure 3.17: Fault detection rate curves for ramp deviations on one of the scores (left) and sensor measurements (right) of the AR(1) process. The fault magnitude is defined as d times the standard deviation.

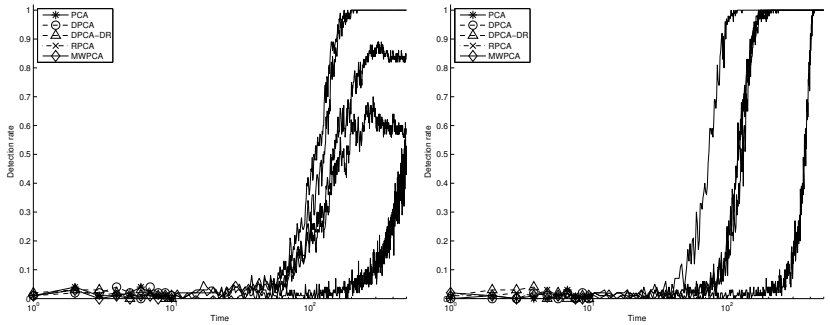

Figure 3.18: Score (left) and sensor (right) fault detection rate on the maximum simulated score and sensor ramp faults for the AR(1) process at each faulty time period averaged over all runs.

3.5.2 MA

In Figures 3.19 and 3.20 we show the detection rates and speed of detection for ramp faults on the MA(1) process. The ranking of the methods by score fault detection is the same as in the step fault case.

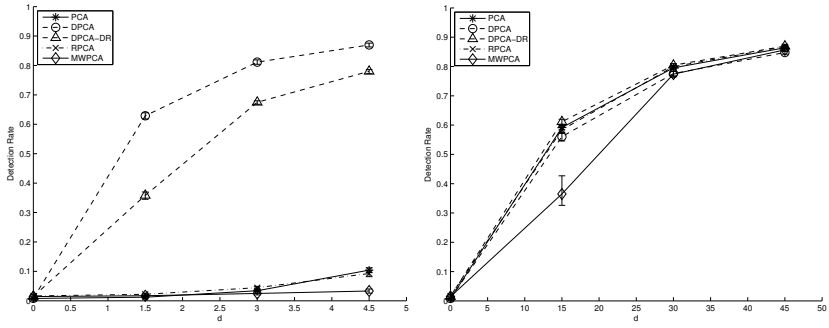

Figure 3.19: Fault detection rate curves for ramp deviations on one of the scores (left) and sensor measurements (right) of the MA(1) process. The fault magnitude is defined as d times the standard deviation.

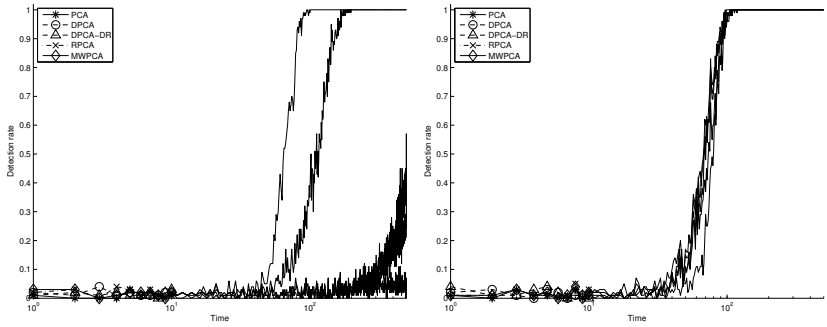

Figure 3.20: Score (left) and sensor (right) fault detection rate on the maximum simulated score and sensor ramp faults for the MA(1) process at each faulty time period averaged over all runs.

3.5.3 ARI

In Figures 3.21 and 3.22 we show the detection rates and speed of detection for ramp faults on the ARI(1,1) process. The ranking of score and sensor fault

detection accuracy is the same as in the step case, but here we see that RPCA and MWPCA have far weaker fault detection capacities since they adapt to the ramp faults.

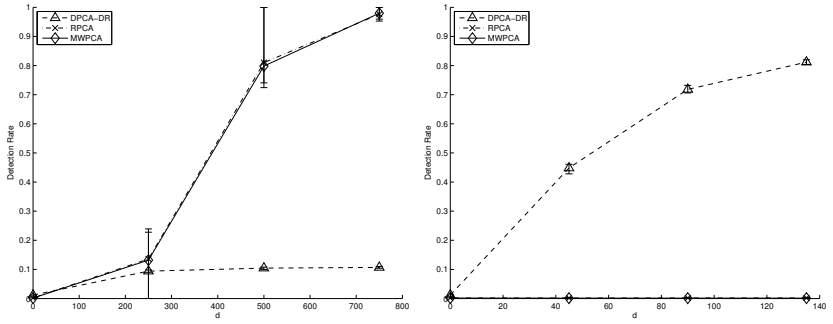

Figure 3.21: Fault detection rate curves for ramp deviations on one of the scores (left) and sensor measurements (right) of the ARI(1,1) process. The fault magnitude is defined as d times the standard deviation.

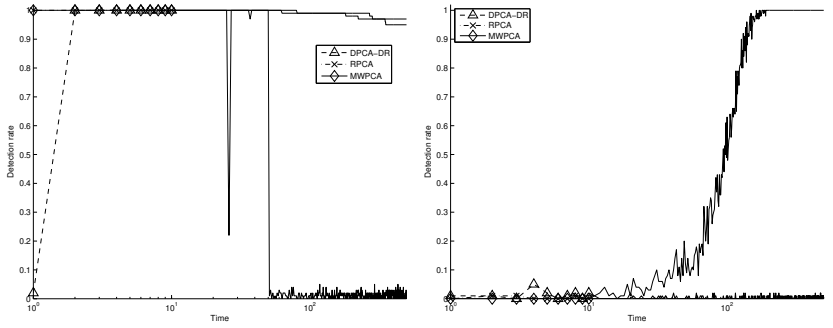

Figure 3.22: Score (left) and sensor (right) fault detection rate on the maximum simulated score and sensor ramp faults for the ARI(1,1) process at each faulty time period averaged over all runs.

3.5.4 IMA

In Figures 3.23 and 3.24 we show the detection rates and speed of detection for ramp faults on the IMA(1,1) process. The ranking of the methods in terms of fault detection is largely consistent with what we saw for the step fault scenario, except that the score fault detection curves of DPCA-DR and RPCA cross. For

the largest score fault magnitude considered, though, RPCA still outperforms DPCA-DR.

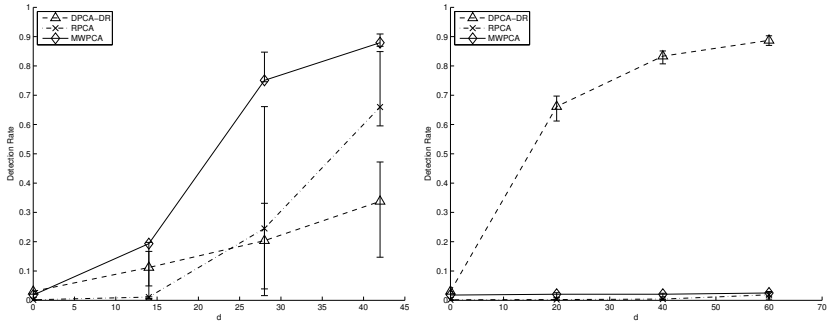

Figure 3.23: Fault detection curves for ramp deviations on one of the scores (left) and sensor measurements (right) of the IMA(1,1) process. The fault magnitude is defined as d times the standard deviation.

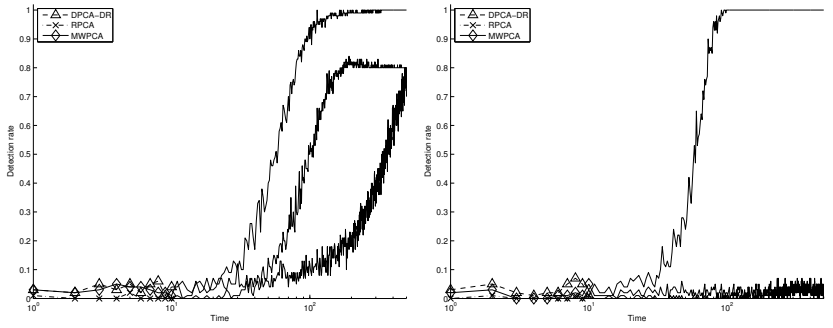

Figure 3.24: Score (left) and sensor (right) fault detection rate on the maximum simulated score and sensor ramp faults for the IMA(1,1) process at each faulty time period averaged over all runs.

3.5.5 NSS

In Figures 3.25 and 3.26 we show the detection rates and speed of detection for ramp faults on the NSS process. As before DPCA, is the best method for detecting score faults, and DPCA-DR is best for detecting sensor faults. Static PCA delivers good performance as well, while the adaptive methods fair poorly.

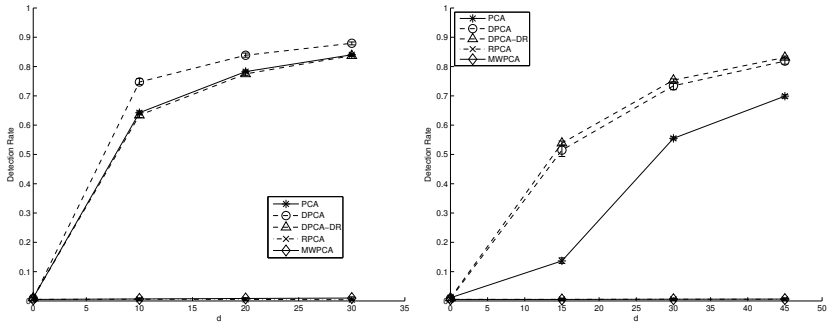

Figure 3.25: Fault detection rate curves for ramp deviations on one of the scores (left) and sensor measurements (right) of the NSS process. The fault magnitude is defined as d times the standard deviation.

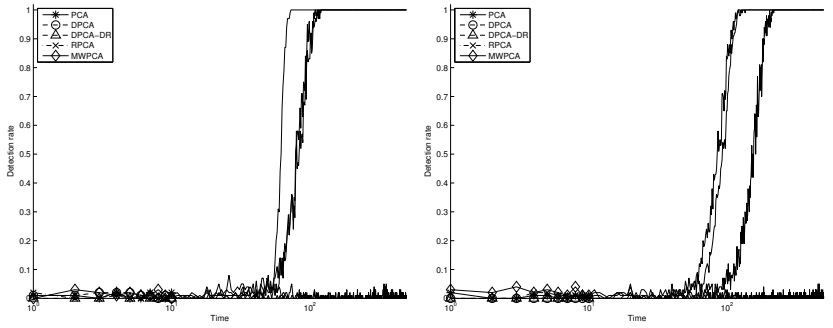

Figure 3.26: Score (left) and sensor (right) fault detection rate on the maximum simulated score and sensor ramp faults for the NSS process at each faulty time period averaged over all runs.

3.6 The Tennessee Eastman process

The simulated processes we have considered thus far have the advantage of being simple and transparent, in the sense that the dynamics of the system and the changes caused by the introduction of faults are readily intelligible. However, these processes are not as complex as those encountered in most real applications. A simulation of the Tennessee Eastman (TE) chemical production process, introduced by Downs and Vogel (1993), provides a more realistic testing environment. We will use the data sets employed by Russell et al. (2000) (available at http://web.mit.edu/braatzgroup), of a controlled version of the Tennessee Eastman process. Twenty-one fault scenarios are considered,

each corresponding to a data set containing 960 observations collected at a sample interval of 3 minutes, with the fault introduced after 8 hours. All the manipulated and measurement variables, except the agitation speed of the reactor's stirrer (which is always constant), are used for monitoring, giving a total of 52 variables. Manipulated variables are controlled input variables. Measurement variables are direct measurements of the process. Table 3.4, summarizes the 21 fault scenarios we will apply the methods to.

Table 3.4: Tennessee Eastman fault types

Fault	Description	Type
IDV(1)	A/C feed ratio, B composition constant (Stream 4)	Step
IDV(2)	B composition, A/C ratio constant (Stream 4)	Step
IDV(3)	D feed temperature (Stream 2)	Step
IDV(4)	Reactor cooling water inlet temperature	Step
IDV(5)	Condenser cooling water inlet temperature	Step
IDV(6)	A feed loss (Stream 1)	Step
IDV(7)	C header pressure loss — reduced availability (Steam 4)	Step
IDV(8)	A, B, C feed composition (Stream 4)	Random variation
IDV(9)	D feed temperature (Stream 2)	Random variation
IDV(10)	C feed temperature (Stream 4)	Random variation
IDV(11)	Reactor cooling water inlet temperature	Random variation
IDV(12)	Condenser cooling water inlet temperature	Random variation
IDV(13)	Reaction kinetics	Slow drift
IDV(14)	Reactor cooling water valve	Sticking
IDV(15)	Condenser cooling water valve	Sticking
IDV(16)	Unknown	
IDV(17)	Unknown	
IDV(18)	Unknown	
IDV(19)	Unknown	
IDV(20)	Unknown	
IDV(21)	The valve for Stream 4 was fixed at the state position	Constant position

Table 3.5 shows the parameter settings of the models we used to monitor the TE process. The high number of retained components is due to the threshold of 95% used by the CPV criterion. Good detection results can also be obtained with lower thresholds though the evaluation of the methods does not yield a different interpretation.

Table 3.5: Parameter settings used to monitor the TE process.

	PCA	DPCA		DPCA-DR		RPCA		MWPCA	
Process	LV	LV	Lags	LV	Lags	LV	η	LV	H
TE	37	195	1–18	195	1–18	35	0.9983	32	163

Table 3.6 shows the fault detection rates of the monitoring statistics of each method across the fault scenarios. Since the TE process is autocorrelated,

but does not exhibit significant non-stationarity, we expect that DPCA and DPCA-DR will be the methods best suited to identifying the faults. In practice, we find that this to be largely the case, with DPCA-DR delivering the highest fault detection across most of the scenarios. If we consider the detection rates of RPCA and MWPCA, we find that these are often much lower than those of the other methods due to adaptation to the fault. Despite the range of methods considered, we also find that none reliably detect faults 3, 9 or 15, which is consistent with the findings of Russell et al. (2000).

Table 3.6: Tennessee Eastman fault detection results. The method giving the best performance is in bold.

Fault	PCA		DPCA		DPCA-DR		RPCA		MWPCA	
	T^2	Q	T^2	Q	T^2_{prev}	T^2_{res}	T^2	Q	T^2	Q
1	0.991	**0.996**	0.989	0.994	0.995	0.994	0.991	0.991	0.994	**0.996**
2	**0.983**	0.976	0.981	0.979	0.976	0.970	**0.983**	0.968	0.981	0.969
3	0.002	**0.010**	0.001	0.007	0.009	0.004	0.001	0.002	0.006	0.004
4	0.105	0.995	0.878	**0.999**	0.993	**0.999**	0.144	0.911	0.006	0.006
5	0.218	0.238	0.236	0.341	**0.999**	**0.999**	0.221	0.099	0.233	0.215
6	0.989	**0.999**	0.984	**0.999**	**0.999**	**0.999**	0.993	**0.999**	0.995	**0.999**
7	**0.999**	0.727	**0.999**	**0.999**	**0.999**	**0.999**	**0.999**	0.700	**0.999**	0.940
8	0.969	0.925	0.968	0.974	**0.975**	0.970	0.971	0.844	0.971	0.968
9	0.001	0.000	0.000	0.002	**0.006**	0.002	0.002	0.000	0.001	0.001
10	0.211	0.346	0.147	0.381	0.876	**0.880**	0.004	0.004	0.016	0.011
11	0.358	0.493	0.662	**0.919**	0.703	0.775	0.077	0.065	0.019	0.016
12	0.979	0.903	0.995	0.996	**0.998**	0.995	0.980	0.865	0.983	0.973
13	0.941	0.948	0.940	0.950	0.956	0.951	0.943	0.940	0.976	**0.998**
14	0.984	0.843	0.998	0.999	0.602	0.995	0.981	0.874	**0.999**	**0.999**
15	0.001	0.007	0.000	0.005	**0.039**	0.021	0.000	0.002	0.010	0.002
16	0.062	0.326	0.055	0.306	**0.891**	0.874	0.000	0.005	0.001	0.004
17	0.753	0.936	0.864	0.970	**0.973**	0.971	0.754	0.891	0.197	0.218
18	0.889	**0.898**	0.888	0.900	**0.898**	**0.898**	0.891	0.895	0.890	0.888
19	0.010	0.030	0.069	0.270	**0.421**	0.397	0.004	0.000	0.007	0.005
20	0.189	0.469	0.255	0.597	**0.819**	0.797	0.102	0.094	0.074	0.060
21	0.332	0.449	0.377	0.422	**0.472**	0.315	0.000	0.000	0.015	0.011

3.7 Discussion

As expected, DPCA and DPCA-DR typically handle autocorrelation well, and RPCA and MWPCA do the same for some kinds of non-stationarity. Interestingly, our results show that static PCA is effective enough when the process exhibits simple dynamics, as in the AR(1) and NSS cases, and when the faults occur in the scores. Nonetheless, for processes with complex dynamics and time-dependent features, neither PCA nor DPCA can cope with the natural fluctuations of the data. DPCA-DR displays a markedly superior capability for detecting sensor measurements faults in all cases, while score faults with low magnitude pass undetected except in the MA(1) and NSS cases. This characteristic is a consequence of DPCA-DR's ability to estimate the current

scores using missing data imputation techniques, as it will be explained later. RPCA and MWPCA do detect score faults, but eventually adapt to them. The expectation had been that the adaptability of these methods would lead to a more accurate local description of the process and higher quality monitoring statistics as a consequence. Therefore, it was revealing to find while performing well for the ARI(1,1) and IMA(1,1) processes, the adaptive methods were less well suited to the NSS process than the non-adaptive methods. Indeed, the non-adaptive methods perform well because they cover the whole period of the process in the calibration phase. On the other hand, the adaptive models 'lag behind' slightly in the sense that the limits and parameters pertain to slightly older local realizations of the process, causing the weaker performance.

In Table 3.7 we summarize our simulation results, identifying poor performance (–), acceptable performance (∘), good performance (+) and the best performing method ($+^*$). In cases where two methods produce essentially identical results, and have the highest detection rates, both are classified as the best. These assessments are purely qualitative, and are meant as guidelines to compliment the quantitative results presented in the simulations section.

Table 3.7: Summary of fault detection performance.

	PCA		DPCA		DPCA-DR		RPCA		MWPCA	
Type	Score	Sensor	Score	Sensor	Score	Sensor	Score	Sensor	Score	Sensor
AR	$+^*$	∘	$+^*$	–	–	$+^*$	+	–	+	–
MA	–	+	$+^*$	+	∘	$+^*$	–	∘	–	∘
ARI	- -	- -	- -	- -	–	$+^*$	$+^*$	–	$+^*$	–
IMA	- -	- -	- -	- -	–	$+^*$	+	–	$+^*$	–
NSS	∘	–	$+^*$	$+^*$	∘	$+^*$	–	–	–	–

Since we have fixed the standard deviations of $\boldsymbol{\varepsilon}_t$, and $\boldsymbol{e}_t$, the difficulty of detecting score and sensor faults is comparable across processes. In Figure 3.27, we plot the log of lowest magnitude fault for which good detection is achieved by the best performing method against the proportional contribution of score and sensor errors to the total variation. In the left plot, the x-axis is $trace(cov(\boldsymbol{\varepsilon}_t))/trace(cov(\boldsymbol{y}_t)) \times 100$, and in the right plot it is $trace(cov(\boldsymbol{e}_t))/trace(cov(\boldsymbol{x}_t)) \times 100$, where $\boldsymbol{y}_t$ and $\boldsymbol{x}_t$ are obtained from a reference data set for each process. We see that this relationship is non-linear, even after applying the log, with the difficulty of detecting faults typically increasing the more dominant the process variation is. This is logical given that the process variance does not grow linearly between processes, and when the process variation is large relative to the fault magnitude it tends to mask small faults. Still MA(1) faults are detected a bit earlier than we might otherwise expect because of the behavior of DPCA described in the section on the MA(1) simulation.

Additional process scenarios should yield further insights. While this may be the case, the results of this investigation highlight the complexity of control

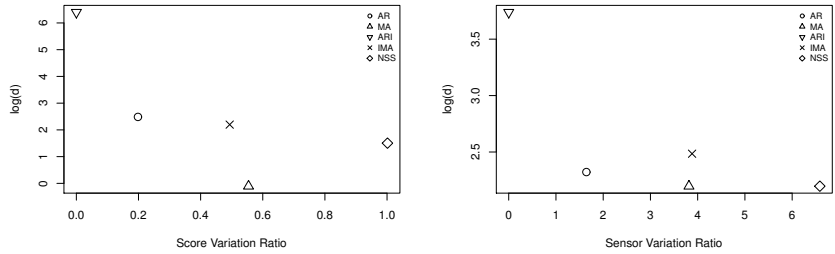

Figure 3.27: Minimum $\log(d)$ for which good fault detection is achieved against relative contribution of score and sensor error to total score and measurement variation.

chart modeling of time-dependent processes; even for the fundamental cases we covered. Even when model parameters are well chosen, methods designed for a process type may be outperformed by more basic methods, as happens in the NSS process. Furthermore, the results also point to the importance of understanding synergies between the monitoring methods, such as we observe in the ARI(1,1) and IMA(1,1) processes. RPCA and MWPCA detect faults in the scores while DPCA-DR is more suitable for the measurements. For this reason, future research into methods more suited for a specific process may be of interest, even if they were not originally designed for that type of process.

In analyzing the fault diagnosis performance of the methods on the TE data, we found that DPCA-DR was generally the most effective method, followed distantly by PCA and DPCA. The poor performance of the adaptive methods indicates that selecting this sort of method should perhaps be avoided unless non-stationarity is of real concern. If non-stationarity is minimal or not expected, it is better to rely on methods which explicitly assume stationarity.

In general, most of the methods display a constant fault detection rate following the introduction of the fault. This is especially true of the non-adaptive methods. As expected, the adaptive methods often report lower detection rates as the time from the fault grows because they adjust to the faulty scenario. More surprisingly, our results reveal that the detection rates of the dynamic methods can increase (as in the MA(1) process) or decrease (as in the IMA(1,1) process) once the lagged observations consist entirely of faulty observations.

An interesting finding of this study was the capability of DPCA-DR to deal with non-stationary processes. Since the studied processes (namely, AR(1), MA(1), ARI(1,1) and IMA(1,1)) can be expressed as a function of past observations, a

regression model with a sufficient amount of past observations is able to explain the process dynamics properly. This modeling is, in fact, conducted by the underlying DPCA model of DPCA-DR. The missing data estimation step also makes it possible to perform a one-step-ahead prediction of the observations and scores that, given information from past observations, best agree with the latent variables subspace. The estimated values will follow the same NOC non-stationary pattern in which the model was trained, and therefore deviations from that structure are reflected on their residuals and captured by the respective monitoring statistics. Surprisingly, these good modeling properties also lead to low detection capabilities when changes occur at the scores level. In this case, only the first faulty observations are signaled as an alarm, since they do not comply with the past observed behavior. As these faulty observations are then used to estimate the subsequent observations, the deviation is no longer observed and no detection is done. A similar complication occurs with DPCA when an atypical observation is obtained. While this atypical observation will be detected correctly, in the next time period, it will be added as the first lag of the next observation. This may lead to the wrong classification of the new observation since the previously atypical observation also has an impact on the scores computation at the current time. Investigation into possible solutions could improve the performance of these methods. Although we used a more accurate method to select the number of lags for DPCA and DPCA-DR, as a sensitivity analysis we also considered the lag selection method of Ku et al. (1995). In most of the processes studied, this method selected no lag for DPCA, making it equivalent to PCA, and revealing the need for the more refined approach.

3.8 Conclusions

This chapter has explored how well a range of methods detect faults when applied to challenging process types. The results are relative performance are an important indicator of how fit these methods are to monitor those process types. However, only one parametrization of each process type was studied. Furthermore, by including faults in the data another crucial and related feature of the methods is obscured; namely how well they model the processes. Chapter 4 will investigate this wider range of modelling considerations.

Chapter 4

Process monitoring capabilities of PCA-based methods

Based on: De Ketelaere, B., Rato, T., Schmitt, E., Hubert, M. (2016). Statistical process monitoring of time-dependent data. *Quality Engineering* 28 (1), 127–142.

4.1 Introduction

Contemporary processes are typically highly automated, with in-line sensor technologies that produce vast amounts of data in a short period of time being the common situation. The result is the availability of large process streams that often display autocorrelation because of the fast sampling schemes relative to the process dynamics (i.e. inertial elements defining the settling time of the process). Additionally, in a substantial part of those real-life processes non-stationarity is an important factor. This scenario of multivariate, time-dependent data is one of the most challenging encountered in Statistical Process Monitoring (SPM), but it is often overlooked, although the separate fields of multivariate SPM and SPM for autocorrelated data have received more attention during the last decade [Woodall and Montgomery (2014); Bersimis et al. (2006); Ferrer (2007, 2014)].

In the previous chapter, the fault detection performance of PCA-based methods was evaluated on challenging process configurations. Although this is a crucial feature to understand for these methods, it is also necessary to study how well they model the process they are used to monitor. This performance feature can be evaluated using their FDR. The difference between the approach of Chapter 3 and this chapter is that in the prior the focus was on fault detection, while in this chapter the focus will be on the false detection rate (FDR). Furthermore, while the previous chapter considered only difficult parametrizations of the processes, the simulations in this chapter will consider a spectrum. Since fault detection and modelling capability are related, the combined results of these two chapters can give a more nuanced overview of how these methods can be expected to perform in practice, compared to the insights they can deliver individually. Ergo, we will discuss the use of PCA, DPCA, RPCA and MWPCA for time-dependent processes, and will focus mainly on the ability of these methods and the derived control charts to describe such data. We will also touch briefly upon a similar approach advocated in Wikstrom et al. (1998), where the use of a classical PCA in combination with multivariate time series modeling of the scores is described.

4.2 Simulated, time-dependent processes with a range of parametrizations

This section can be seen as an extension of the work done in Chapter 3. There, time-dependent processes were simulated using aggressive parametrizations that made it difficult to model them. The purpose of that approach was to push the monitoring methods to their limits and reveal behaviors *fault detection* that are not discussed in the literature. In this section, we explore a range of parametrizations, from easy to hard, so that we can identify when the modelling methods accurately model the process, when they show strain, and when they fail. This will be done using the false discovery rate.

Typically, monitoring methods are evaluated for their fault detection in the literature, but good fault detection is predicated on a good model of the process and a correct definition of the related control limits. However, classic monitoring approaches do not ensure that an appropriate model is obtained for a broad range of process dynamics that are typical for real-life applications. Therefore, in this section we evaluate their validity through investigation of the modelling accuracy of the PCA-based methods on the AR(1) and ARI(1,1) processes. The AR(1) is chosen as it is a widely encountered process dynamic in modern processes, and its integrated form (ARI) is used as its nonstationary counterpart.

Following convention (e.g. Burnham et al. (1999); Choi et al. (2006)) we generate data at the subspace level so that we can explicitly control the features monitored by the PCA-based models. To obtain each observation at time t we began by generating five latent variables, $\boldsymbol{y}_t$, according to the equation of the desired AR(1) or ARI(1,1) process. For all processes, we introduce variation onto the process dynamics through $\boldsymbol{\varepsilon}_t \sim \mathcal{N}(\mathbf{0}_5, 0.01\boldsymbol{I}_5)$, where $\boldsymbol{I}_5$ is the 5×5 identity matrix. These are then transformed into a 50-dimensional dataset of measurements computed as

$$\boldsymbol{x}_t = \boldsymbol{P}_0 \boldsymbol{y}_t + \boldsymbol{e}_t, \tag{4.1}$$

where $\boldsymbol{P}_0$ is a 50×5 matrix with orthogonal columns randomly generated once and kept constant for all simulation runs. The $\boldsymbol{e}_t$ are 50×1 vectors of white noise errors, distributed as $\mathcal{N}(\mathbf{0}_{50}, 0.000025\boldsymbol{I}_{50})$, that simulate measurement noise, as is done, for instance, in (Ku et al. (1995) and Lakshminarayanan et al. (1997)). The $\boldsymbol{e}_t$ can be seen as the error at the sensor level, and are set to a small value here under the assumption that sensors are typically reliable. For all methods and simulations, an arbitrary but common CPV of 95% is used.

The AR process is investigated because it is a particularly relevant process type seen the high sampling rate of many contemporary sensors inherently introducing (positive) autocorrelation into the data. Besides being a common process type in real life situations, AR processes have a natural relevance for studying the properties of DPCA.

The AR(1) process is defined in Equation 3.3. We consider values of ϕ equal to 0, 0.1, 0.3, 0.5, 0.7, 0.9, with larger values of the parameter corresponding to stronger autocorrelation. Setting $\phi = 0$ gives us a process with i.i.d. observations, which is the reference condition for which the assumptions of PCA and the theoretical control limits defined above are valid.

The ARI(1,1) process is defined in Equation 3.5. The ARI(1,1) process is considered because the adaptive methods were designed to address nonstationary processes.

Figure 4.1 depicts the AR(1) and ARI(1,1) processes with ϕ values of 0.1 and 0.9. In general, when ϕ increases, the variance of $\boldsymbol{x}_t$ increases (with a factor $(1-\phi^2)^{-1}$ in case of the AR(1)), so decreasing the relative effect of the noise. Also the unit root introduced in the nonstationary processes has a marked influence on the total signal variance, being even more pronounced than the effect of ϕ. Because of both effects, different scaling factors were required in Figure 4.1 to visualize the typical behavior of the different simulation settings. Moving from left to right and top to bottom, the scaling factors used were 10, 5, 0.5 and 0.025. As a result, while the ARI(1,1) with $\phi = 0.9$ appears to be relatively well behaved, its own scale is much greater than that of the other processes.

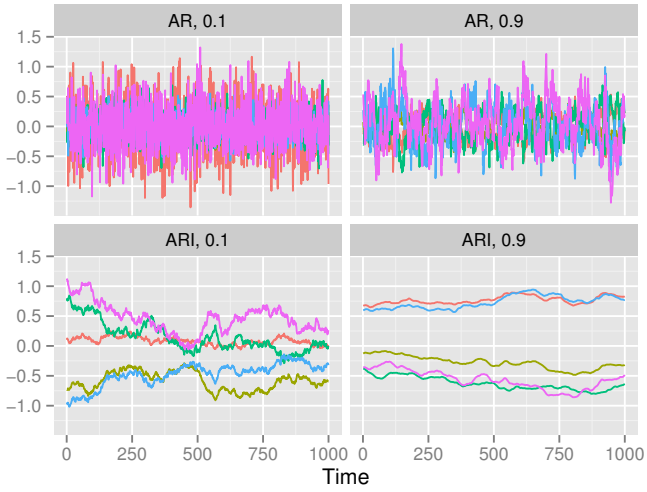

Figure 4.1: Plots of the rescaled AR(1) and ARI(1,1) processes with ϕ values of 0.1 and 0.9.

To assess the performance of the PCA-based models and corresponding control limits, 100 replicates of normal operation conditions (NOC) were generated. Each of these replicates is composed by 7000 NOC observations, divided into a calibration (first 6000 observations) and test (last 1000 observations) dataset. Models were specified for each replicate using the respective calibration dataset and their performances were subsequently assessed on the contiguous test dataset. False detection rates (FDR) were computed for each replicate as the number of observations above the theoretical control limit divided by the total number of observations in the test phase. Therefore, for each process type and ϕ, 100 FDRs were obtained. The distribution of the observed FDRs is then considered as a measurement of the models performance. For DPCA, RPCA and MWPCA additional parameters need to be chosen, such as the number of lags l, the forgetting parameter η and the window length H. In order to do so, an additional calibration dataset with 5000 NOC observations was generated for each combination of process type (AR(1) or ARI(1,1)) and ϕ value. The number of lags used by the DPCA method was selected using the method of Rato and Reis (2013b).

Although the selection of the additional parameters for adaptive methods is critical to their proper implementation, this topic is not well covered in the literature. We based their choice on evaluating a range of possible values and

assessing their appropriateness. The minimum and maximum values for η and H are $[0.9, 0.9999]$ and $[50, 2500]$, respectively. For each process type models with these candidate parametrizations were applied to the additional calibration dataset, after splitting it into two equal parts. The first part is used to set up the models with the given values of η and H, and the second part is then used to calculate the Sum of Squared Prediction Errors (SSPE) following the suggestions of Schmitt et al. (2016) (later developed in Chapter 5). The η and H values that minimizes the SSPE are then employed for process modelling. This approach can be thought of as a generalization of choosing the weighing factor in an EWMA control chart [Montgomery (2008)].

4.2.0.1 Simulation Results: AR(1)

For each of the simulation settings we considered, the parameter selection procedure explained above resulted in the parametrization which is given in Table 4.1. One trend that is apparent in this table is that the number of retained latent variables tends to decrease to the correct number, five, as ϕ increases. The fact that the i.i.d. case did not lead to the underlying five latent variables is mainly due to the relatively large influence of the noise in these stationary cases and the chosen CPV value of 95%. The influence of the noise through $\boldsymbol{e}_t$ is lowered when the autocorrelation increases (see Figure 4.1), explaining why, for higher values of ϕ, the correct number of latent variables is extracted.

The impact of the dynamic features of the data is also visible on the lag selection procedures. In particular, the Ku et al. (1995) method selects zero lags for all AR(1) processes, except for $\phi = 0.9$, which has one lag (results not shown for the sake of brevity). Thus, this lag selection methodology finds that the dynamic relationships are not significant when the process exhibits moderate dynamics. This result is in line with the findings of Rato and Reis (2013b), who also concluded that the Ku et al. (1995) method has a tendency to underestimate the true dynamics of the data. As mentioned before, to overcome this issue, we present results for DPCA where the lag selection procedure of Rato and Reis (2013b) is implemented. In this approach the number of lags is not necessarily the same for all variables. For the cases study considered, the maximum number of lags was consistently set as one, while the effective lag of each variable varied between zero and one. This means that some variables do not require any lag in order to describe the process data. Since DPCA also models the cross-correlation structure of the original as well as the lagged variables, the exclusion of redundant lags leads to more parsimonious models. It is noted, however, that in the i.i.d. case ($\phi = 0$), the Rato and Reis (2013b) method on average adds one lag which is undesired. This happens because the optimization algorithm assesses the modelling improvements of consecutive lag structures and

since the zero lag scenario is the first possible structure, it cannot be compared against a previous reference. Subsequently, the lowest feasible lag structure has at least one lag. Nevertheless, this situation can be avoided through further analysis of the data and decision graphs produced by the algorithm.

For the adaptive models, the selected forgetting parameters are all high, indicating that nonstationarity is not dominant in the data. Most of the values are at their upper bound, except for the large ϕ cases. This does not come as a surprise since in case of large ϕ the process mean does deviate from 0 for longer time periods, and the adaptive models try to capture these (random) dynamics by forgetting older observations faster.

Table 4.1: Parameter settings for monitoring methods in the AR(1) processes. Ranges are given for variable parameters, with the most frequent value in brackets.

	PCA	DPCA		RPCA		MWPCA	
ϕ	k	k	Lags	k	η	k	H
0	7	14	0-1 (1)	7-8 (7)	0.9999	6-8 (7)	2500
0.1	8	14	0-1 (1)	6-8 (7)	0.9982	6-8 (7)	2500
0.3	6	12	0-1 (1)	6-7 (7)	0.9999	6-7 (7)	2469
0.5	5	10	0-1 (1)	5	0.9999	5-6 (5)	2496
0.7	5	10	0-1 (1)	5	0.9998	5	1775
0.9	5	7	0-1 (1)	5	0.9981	5	886

Monitoring was performed on each of the AR(1) settings and the false detection rates (FDR) of the Hotelling's T^2 and Q statistics were recorded. The desired overall FDR is set at 1%, and since we have no knowledge about the correlation between the T^2 and Q statistic, it is assumed to be zero and the Bonferroni correction is applied such that $\alpha_{T^2} = \alpha_Q = FDR/2 = 0.005$. Boxplots of the FDRs for the Hotelling's T^2 and the Q statistics are presented in Figures 4.2-4.5, as a function of the autocorrelation parameter ϕ.

Across the results, we see that the effect of autocorrelation on the modelling properties of the PCA-based methodologies is not strong, except for high values of ϕ. This effect has a greater influence in the Hotelling's T^2 statistic dynamics since the original autocorrelation of the data is directly translated to the scores, which ultimately compromises the reliability of theoretical control limits [Kruger et al. (2004); Vanhatalo and Kulachi (2015)]. Although the observed false detection rate of the Hotelling's T^2 statistics is generally within expectation, we see that the dispersion of the FDR values increases as the autocorrelation increases. This is a direct result of the inherent dynamics on the Hotelling's T^2 statistics, since it increases the probability of having consecutive measurements with similar values. Thus, for replicates where the process experiences sustained deviations from the model (i.e., high values of Hotelling's T^2), the false detection rate is higher than specified, while the

converse happens when the process runs close to the model. In that case, the Hotelling's T^2 statistic exhibits consecutive, low values. This results in more variable detection performance, even though the average FDR is close to the desired value. Although the FDRs obtained for the T^2 are generally in line with expectations, extensions to the PCA framework can produce some additional improvement. This was the idea behind the approach of Wikstrom et al. (1998), which applies an ARIMA modelling approach to the scores. We indeed observed a modest decrease in this dispersion for high values of ϕ, but since the Wikström approach does not consider the residual space, it cannot solve the problem seen with the Q statistics.

On the other hand, since the Q statistic is related with the model residuals, it should be serially decorrelated as long as the appropriate number of latent variables is retained. This in turn should lead to good monitoring performance. We observe that this is the case for low values of ϕ, since reasonable FDR are obtained. For larger values of ϕ the models also produced serial decorrelated residuals. However, the scores subspace is not accurately explaining the dynamic characteristics of the data, causing the residuals to be greater than expected, which leads to a higher FDR than the target. While extensions based on classical time series methods, such as that of Wikstrom et al. (1998), are applicable to the T^2 statistic as we mentioned above, the number of variables used to calculate the Q-statistic can be extremely large, and therefore beyond the capacity of such methods.

By using the correct number of lags in the DPCA model, an elimination or at least reduction of some of the misspecifications is expected, resulting in an improved modelling and monitoring performance. However, even though the DPCA approach is able to follow the process dynamics more closely, it still produces monitoring statistics (especially the Hotelling's T^2) with dynamic characteristics. This indicates that DPCA is prone to the same deficiencies identified earlier for PCA, which essentially lead to misspecified control limits.

The adaptive methods show similar results as the non-adaptive PCA and DPCA methods. The reason for this are the very large forgetting parameters η and window size H presented in Table 4.1. These cause the methods to forget only very slowly. These large values indicate that nonstationarity is not a major issue for this process, which in fact is correct. For MWPCA, in case of $\phi = 0.9$, the window size H was substantially smaller, causing the FDR dispersion to be substantially larger for the Hotelling's T^2 statistic.

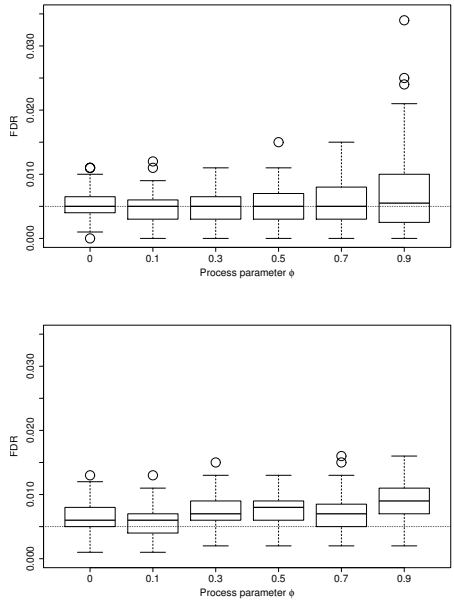

Figure 4.2: False detection rates of the T^2 (top) and Q statistics (bottom) of PCA on the AR(1) process with ϕ ranging from 0 to 0.9. The value of $\alpha_{T^2} = \alpha_Q = 0.005$.

4.2.0.2 Simulation Results: ARI(1,1)

Next, we consider the ARI(1,1) process, again setting the target FDR equal to 1%. The ARI(1,1) process poses a greater monitoring challenge for PCA and DPCA than the AR(1) process because of the apparent nonstationarity (see Figure 4.1). This is expected since neither of these methods is designed to cope with nonstationary behavior. Experiments confirmed that FDRs typically reach 100% for these methods regardless of the value of ϕ, so results are not shown. This is caused by the fact that the data used to build the models are not representative for new data encountered in the test dataset. The approach of Wikstrom et al. (1998) does fail as well when an ARMA model is fitted through the T^2 statistics because they are also nonstationary, so that differencing the scores is required. Furthermore, the nonstationarity around the PCA model remains unexplained by the Wikström approach.

The model parametrizations of the adaptive methods are shown in Table 4.2.

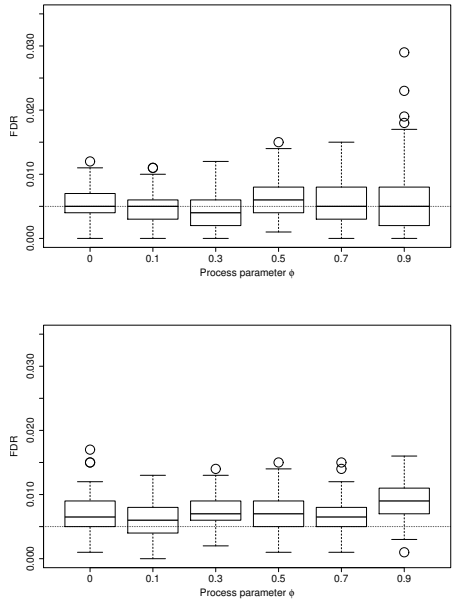

Figure 4.3: False detection rates of the T^2 (top) and Q statistics (bottom) of DPCA on the AR(1) process with ϕ ranging from 0 to 0.9. The value of $\alpha_{T^2} = \alpha_Q = 0.005$.

We can see that in general, as ϕ increases, the forgetting factor decreases, although there are deviations from this pattern. This was to be expected since those situations are dominated by the strongest nonstationarity as we have demonstrated in Figure 4.1, and is in line with the observation for the AR(1) case. RPCA and MWPCA are expected to perform acceptably in this setting since they are able to adapt to process changes. However, both methods produce unacceptable results across all values of ϕ when the theoretical control limits are used in combination with $\alpha = 0.01$. Changing the forgetting factors to improve results did not lead to consistently on-target performance.

The reason for this poor behavior is two-fold. First, when applying RPCA and MWPCA, the models are only updated when a new point is considered in control. When the forgetting factor is not chosen ideally, or when the dynamics of the underlying process change, the adaptive methods can fail to follow those dynamics, leading the model to consider a large portion of the data to be out of control. As stressed before, the right choice of the forgetting factor and its

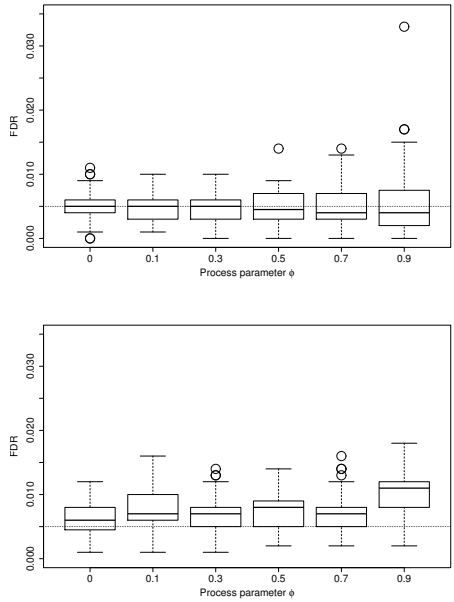

Figure 4.4: False detection rates of the T^2 (top) and Q statistics (bottom) of RPCA on the AR(1) process with ϕ ranging from 0 to 0.9. The value of $\alpha_{T^2} = \alpha_Q = 0.005$.

Table 4.2: Parameter settings for monitoring methods in the ARI(1,1) processes. Ranges are given for variable parameters, with the most frequent value in brackets.

	RPCA		MWPCA	
ϕ	k	η	k	H
0	3-5 (4)	0.9981	3-5 (4)	2500
0.1	3-5 (4)	0.9986	3-5 (4)	1400
0.3	3-5 (4)	0.9972	4-5(4)	700
0.5	3-5 (4)	0.9955	3-5 (4)	450
0.7	3-5 (4)	0.9800	3-5 (4)	250
0.9	3-4 (4)	0.9500	2-4 (3)	100

eventual updating to account for changing dynamics is important, but references are scarce (e.g. Choi et al. (2006)) and the topic deserves further attention.

The second reason for the excessive FDR comes from the fact that the underlying assumptions of the analytical Hotelling's T^2 and Q limits defined earlier and applied here with α set at 1% do not hold. This misspecification causes the

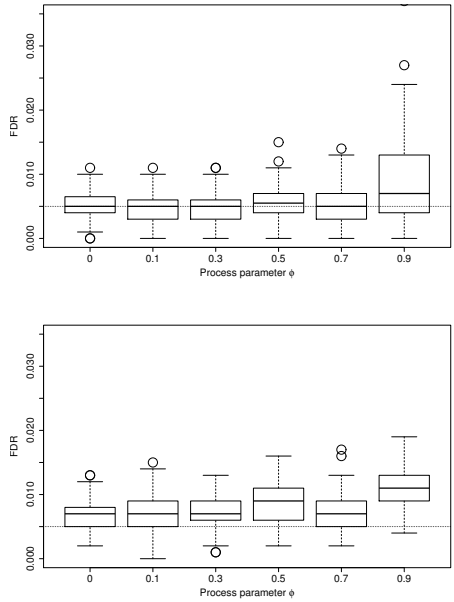

Figure 4.5: False detection rates of the T^2 (top) and Q statistics (bottom) of MWPCA on the AR(1) process with ϕ ranging from 0 to 0.9. The value of $\alpha_{T^2} = \alpha_Q = 0.005$.

control limits to be too tight, so that a substantial number of observations are considered outlying. This in turn prevents the model from being updated since such OOC points are not used for adapting the model. As advocated in Rato et al. (2016), in such cases it is better to tune the α value such that the desired FDR is obtained (for other examples of approaches for adjusting the limits, cf. Ramaker et al. (2005), Camacho et al. (2009), and van Sprang et al. (2002)). In Figures 4.6 and 4.7, we consider smaller values of α_{T^2} and α_Q (see Table 4.3), resulting in higher control limits but acceptable FDR rates. These α values were determined manually by dividing a fixed reference data set in two parts, fitting a model to the first part (5000 observations), and assessing its performance on the second part (15000 observations). Then, the selected α values were applied to all 100 simulation runs. The variable performance in the simulations shows that this approach, while generally effective, does not result in models that generalize to all of the realizations of the process encountered in the simulations. The figures demonstrate that for a substantial number of the simulation runs the FDR is far from the target FDR, with values that reach 100%. The reason

for those extreme cases is that models are only updated when a new point is considered in control. If at the start of the monitoring phase points are considered OOC the model does not update, increasing the probability that later measurements will be considered OOC as well when the process further deviates from the model because of the nonstationarity. Table 4.3 lists the median FDR values since they are not clearly visible in the boxplots. For the T^2 values, the median is actually zero, meaning that no NOC points were considered to be outlying. This illustrates that tuning α for the Hotelling's T^2 statistic is not working adequately and improvements are needed. This is also visualized in Figure 4.6, where indeed the T^2 control limit is too high because of the substantial autocorrelation present in the Hotelling's T^2 statistics.

For the Q statistic, the medians are in line with expectations, meaning that if the method is actively monitoring, the tuned control limit for the Q is adequate. This is illustrated in Figure 4.8, where the Q statistic shows a random behavior in periods of similar limits, and where the amount of adaptation of the control limit in periods of higher/lower residuals is acceptable.

As mentioned above, the tuned α values in Table 4.3 were selected by trial an error adjustment on a validation dataset because no better approach is currently available. From these results we can observe a direct relationship between the value of ϕ and the values of α needed to have a FDR that is in line with expectations: the higher ϕ, the smaller the α values must be. The cause of this relationship is that increasing ϕ increases the process variance, and this increased process variance is not accounted for in the theoretical limits. Therefore, for processes with only moderate nonstationarity, the problems we observe in this simulation may not arise to the same extent, and results listed here can be considered as extremes (worst case scenario). We note that the interquartile range does not show a clear trend except that it is typically larger for the T^2-statistic than for the Q-statistic. It might be that even further refined model parametrizations or additional simulation runs are required for a more obvious pattern to emerge.

A visual appreciation of the monitoring behavior of RPCA applied to an ARI(1,1) process with $\phi = 0.9$ and tuned α values is given in Figure 4.8. From this figure we conclude that the monitoring statistics still show evidence that the model is not completely explaining the structure of the process, which is especially visible in the T^2 statistic. In fact, the recursive nature of RPCA does allow to cover the simple case of nonstationarity, but does not seem to cope with the AR component that is added to it.

Even though this parametrization reduces the detection of small faults, the model is adapting and actively monitoring. Therefore, if the process of interest displays large faults, these methods may still be suitable for monitoring.

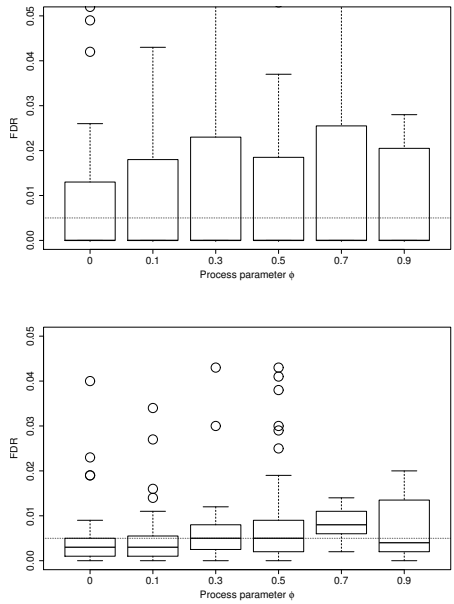

Figure 4.6: False detection rates of the T^2 (top) and Q statistics (bottom) of RPCA on the ARI(1,1) process with ϕ ranging from 0 to 0.9, using the tuned values of α in Table 4.3.

4.3 Discussion

Based on simulation results we covered different forms of time-dependency in process monitoring, focusing on the simple yet challenging cases of an AR(1) and an ARI(1,1) because those types of dynamics are believed to be often present in modern process data.

The results of the AR(1) simulations demonstrate that under moderate dynamics, all of the studied PCA-based methodologies have a similar, acceptable modelling performance. This happens because the optimal parameters of DPCA, RPCA and MWPCA tend to reduce them to static PCA. It is also visible that when process dynamics become more relevant (say, for $\phi \geqslant 0.7$) the models tend to deviate more from expectation, with especially the Hotelling's T^2 statistic to be less reliable, confirming recent results from Vanhatalo and Kulachi (2015). However, simple AR(1) dynamics do not severely compromise the modelling capabilities of the procedures, which are still producing false detection rates

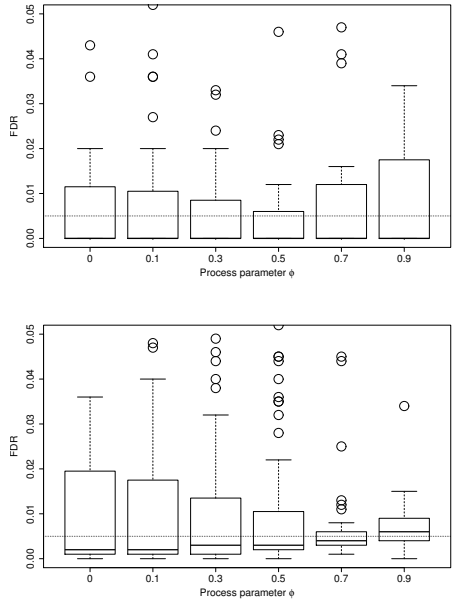

Figure 4.7: False detection rates of the T^2 (top) and Q statistics (bottom) of MWPCA on the ARI(1,1) process with ϕ ranging from 0 to 0.9, using the tuned values of α in Table 4.3.

within expectation.

On the other hand, the ARI(1,1) simulations showed that PCA and DPCA cannot cope with nonstationarity. RPCA and MWPCA, the adaptive methods that are devised for handling nonstationarity, do allow for modeling such data, but plugging the classical values for α in the control limits for the Hotelling's T^2 and Q statistic resulted in FDR values that were unacceptably high. Since no literature is available for defining those control limits under the nonstationarity assumption, it was proposed to relax the control limits for both statistics by searching for α values that result in acceptable FDRs so that these models could continue to adapt to the time-varying process and might be able to detect severe faults. The observation that there is a clear link between the process dynamics and the monitoring method capable of handling such data is partly in line with the results of Camacho et al. (2009), who acknowledge that it is important to reflect the time-varying nature of the process in the model of the SPM method used. Interestingly, Camacho et al. (2009) mention the fact that besides the process dynamics also the fault type to be detected is important when deciding

Table 4.3: Tuned α values for monitoring methods in the ARI(1,1) processes with an intended overall FDR of 0.01 ($FDR_{T^2} = FDR_Q = 0.005$). The observed FDRs are summarized by their median and interquartile range.

		RPCA		MWPCA	
ϕ	Stat.	α	FDR	α	FDR
0	T^2	2×10^{-2}	0 (0.013)	5×10^{-4}	0 (0.012)
	Q	10^{-3}	0.003 (0.004)	5×10^{-4}	0.002 (0.019)
0.1	T^2	2×10^{-2}	0 (0.018)	1.5×10^{-4}	0 (0.011)
	Q	1.5×10^{-3}	0.003 (0.005)	10^{-4}	0.003 (0.017)
0.3	T^2	2×10^{-2}	0 (0.023)	5×10^{-5}	0 (0.009)
	Q	1.5×10^{-3}	0.005 (0.007)	5×10^{-5}	0.003 (0.013)
0.5	T^2	1.5×10^{-2}	0 (0.019)	6×10^{-5}	0 (0.006)
	Q	10^{-3}	0.005 (0.007)	10^{-5}	0.003 (0.009)
0.7	T^2	1.2×10^{-2}	0 (0.026)	2×10^{-5}	0 (0.012)
	Q	1.5×10^{-5}	0.008 (0.005)	10^{-7}	0.004 (0.003)
0.9	T^2	1.7×10^{-2}	0 (0.021)	1.5×10^{-6}	0 (0.018)
	Q	5×10^{-6}	0.004 (0.012)	10^{-9}	0.006 (0.005)

on the best monitoring method. This is ultimately true, and fault detection is probably the most important aspect when considering SPM methods. As an example, Rato et al. (2016) concluded that the capability of the adaptive methods to detect ramp faults is highly dependent on the forgetting factor chosen, and should be considered carefully.

The nonstationarity as introduced into the simulations are extreme cases, as can be seen from Figure 4.1. Performance may be more appropriate on processes with less severe forms of nonstationarity, like processes showing mild nonstationarity or simple drifts due to sensor aging. In those cases, the proposed extensions to PCA might be able to capture the drifts thus describing the data adequately. Contrarily, in order to turn these models into valid and powerful monitoring schemes work is required into a proper definition of the control limits connected to the methods. We believe that this direction of research is highly relevant. In case of more complex nonstationary behavior, work is required into the modeling aspect as we demonstrated in the ARI(1,1) case and high ϕ values where even after adapting α deviating FDR values were noted.

Although nonstationarity is present in a wide range of processes, the use of

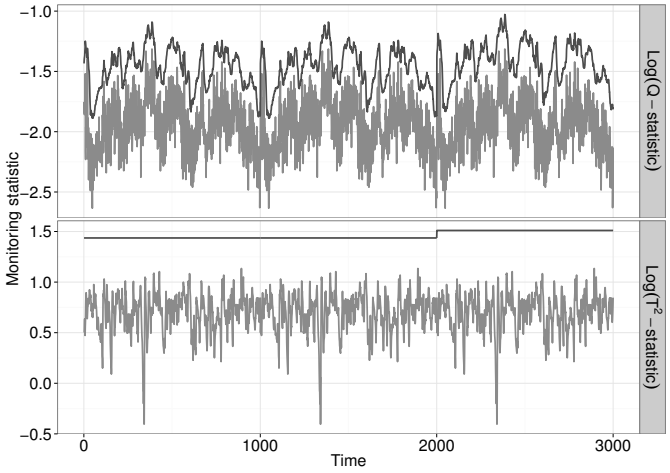

Figure 4.8: RPCA-based control chart for the ARI(1,1) process with ϕ equal to 0.9. The value of $\alpha_{T^2} = 1.7 \times 10^{-2}$ and $\alpha_Q = 5 \times 10^{-6}$. The control statistics are in log_{10} to improve readability.

adaptive models is still limited, especially in the multivariate case. The moderate results we have shown are only a partly explanation. From the practice side, the lack of intuition with the methods by the process owners themselves (often engineers) is a barrier as well. From that perspective, it could be advantageous to translate the parameter choice of e.g. the RPCA into a selection procedure for parameters engineers are used to. More specifically, engineers typically have a good idea of the process dynamics in terms of their in control *frequency spectrum*, i.e. the speed of change which is typical to those processes. This behavior can be visualized through the generation of the Power Spectral Density (PSD) of the process, denoting the power of the signal as a function of the frequency (Oppenheim and Schafer (1975)). Typically, only the slow dynamics are proper to the underlying process, so that the SPM scheme should apply a low pass filtering of the data. In essence, the cut-off frequency determining the frequencies which do (not) pass the SPM model are well related to the AR and/or MA terms. Said this, we feel that bridging the gap between the engineering and statistical reasoning could help implementation of adaptive SPM methods. This implementation issue is the last barrier: software to cope with such multivariate SPM models is not wide-spread, and is required to translate advanced multivariate SPM from a pure research field into a practical solution.

4.4 Conclusions

In this paper we have highlighted the variety of challenges posed by time-dependent processes. We showed that monitoring high-dimensional processes with autocorrelation can be successfully achieved using PCA-based methods. However, for the ARI(1,1) processes, we found that even methods which are purportedly designed to address non-stationarity (RPCA and MWPCA) have difficulties in specifying a suitable model for that process type. Extensions to RPCA and MWPCA exist in the literature (for examples, see Chapter 2) that may offer improvements over the basic implementations. However, these methods have not been thoroughly compared in the literature, so it is difficult to recommend one in particular.

Chapter 5

Parameter selection guidelines for adaptive PCA-based control charts

Based on: Schmitt, E., Rato, T., Reis, M., De Ketelaere, B., Hubert, M. (2016). Parameter selection guidelines for adaptive PCA-based control charts. *Journal of Chemometrics*, 30 (4), 163-176.

5.1 Introduction

The need to provide accurate Statistical Process Monitoring (SPM) of high-dimensional processes arises in disciplines as varied as health care, industry, information technology, economy, and precision agriculture. Principal Component Analysis (PCA)-based control charts are widely used for such task since they are well-suited for modelling this type of data [Kourti (2005) and Chapters 2 and 3]. This type of control chart usually assumes i.i.d. conditions. However, in many applications, causes such as small changes in input materials or uncontrollable conditions introduce non-stationarity in the system. These changes do not constitute faults, but a monitoring model fitted on earlier observations may cease to be representative of the process, leading to a high false detection rate (FDR). In the context of PCA-based control charts, Recursive PCA (RPCA) and Moving Window PCA (MWPCA) were proposed to overcome

this problem by updating the PCA monitoring model with new, in-control observations as they are encountered.

RPCA exponentially downweights the influence of older observations, while MWPCA constructs a model based on observations contained in a time window ending in the current period (the similarity between these methods that was touch upon briefly in Section 2.6.4 will be explored in simulation results). These methods and their extensions have proven useful in monitoring non-stationary processes, but the selection of the parameter dictating how older observations are forgotten remains an open problem. The most widely implemented (most likely due to its simplicity) parametrization is that of a constant forgetting parameter, yet there remains a lack of guidelines on how to select it. Frequently, the topic is left unmentioned or practitioners are referred to expert knowledge. In Section 5.2.1, we will provide guidelines for how to select this parameter for RPCA and MWPCA.

The key to obtaining an informative control chart is to model the normal operating conditions of a process well, since ideally this results in i.i.d. monitoring statistics when there are no faults. Adaptive PCA-based control charts exhibit monitoring statistics with better characteristics than non-adaptive methods when monitoring time-dependent processes. However, they may still not perfectly model highly time-dependent processes, since they are linear approximations of the process, and do not model it as a whole, like a physical model. As a consequence, the statistics they return may not exactly follow the theoretical behavior assumed by most analytical expressions for the control limits, especially in the scores subspace [Chapter 4]. This problem is not unique to adaptive methods. Non-adaptive methods, such as the basic PCA control chart, may have statistics with non-i.i.d. behavior. Empirical limits based on the FDR on calibration data have been used to partially correct this issue [Russell et al. (2000); Ramaker et al. (2005)]. This is not applicable to adaptive methods because the characteristics of the components used by the PCA model may change, resulting in a large difference between the behavior of the statistics during the calibration phase and subsequent time periods that simple empirical limits do not account for. In Section 5.2.2, we propose an approach for selecting control limit parameters for adaptive methods using a combination of analytical and empirical methods. After providing guidelines for parameter selection, we illustrate monitoring results on a simulation in Section 5.3 and two real data examples in Section 5.4. Finally, we discuss our findings in Section 5.5 and conclude in Section 5.6.

5.2 Parameter selection

Taking notation from Chapter 2, we will denote an adaptive PCA model at time t as $m_t(\bar{\boldsymbol{x}}_t, \boldsymbol{P}_{t,k}, \delta, \boldsymbol{\alpha})$, where δ is a forgetting parameter (η in the case of RPCA, and H in the case of MWPCA), $\bar{\boldsymbol{x}}_t$ and $\boldsymbol{P}_{t,k}$ are the weighted center and the loadings matrix based on a covariance matrix, and $\boldsymbol{\alpha}$ is the vector with elements α_{T^2} and α_Q. The approach makes use of a *normal operating conditions* (NOC) data set $\boldsymbol{X}$ that can be partitioned into two data sets, one for calibration $\boldsymbol{X}_c = \{\boldsymbol{x}_t,\ t = 1, \ldots, C\}$ and one for validation $\boldsymbol{X}_v = \{\boldsymbol{x}_t,\ t = C+1, \ldots, C+V\}$, containing C and V observations respectively. As far as we know, there is no general guidance on how to split $\boldsymbol{X}$ as it depends a lot on the abundance and the dynamics of the data. At least both the calibration and validation set should be large enough to represent the dynamic behavior of the process. With larger data sets, such as in simulations or some industrial applications, the splitting can then be rather generous (50%-50%) or even less for the calibration set in order to speed up the construction of the models. With smaller data sets, the splitting must be such that enough data is used to fit a reasonable NOC model, hence a larger proportion of calibration data might be needed.

The goal is to identify the parameter values δ^* and $\boldsymbol{\alpha}^*$, the values of the forgetting parameter and $\boldsymbol{\alpha}$ that will be used in the final monitoring model. To do so, M candidate models are initialized on the calibration data, $m_C^j(\bar{\boldsymbol{x}}_C, \boldsymbol{P}_{C,k}, \delta, \boldsymbol{\alpha})$ for $j = 1, \ldots, M$, using candidate values of δ or $\boldsymbol{\alpha}$ (Sections 5.2.1 and 5.2.2 show how selection of the two parameters can be separated). In the case of MWPCA, the last H observations in the training data are used to construct each model. In the case of RPCA, a PCA model is initialized on the first $k+1$ observations in the training data, then carried forward to the end of the training period using the appropriate value of η. Then, monitoring is carried out on $\boldsymbol{X}_v$, consecutively evaluating observations in $\boldsymbol{X}_v$ and updating $\bar{\boldsymbol{x}}_t$ and $\boldsymbol{P}_{t,k}$ until the model is $m_{V+C}^j(\bar{\boldsymbol{x}}_{V+C}, \boldsymbol{P}_{V+C,k}, \delta, \boldsymbol{\alpha})$. In the following sections, we show how δ^* and $\boldsymbol{\alpha}^*$ are selected given the output from these validation runs.

5.2.1 Determining the forgetting parameter

RPCA and MWPCA both require the selection of a forgetting parameter; η and H. A good choice should properly account for the level of non-stationarity in the data and lead to an accurate model. A similar problem is the selection of the forgetting parameter in the EWMA and MEWMA control charts. For those cases, Pfeffermann and Allon (1989) and Montgomery (2008) recommend selecting the value of the forgetting parameter so that the Sum of Squared

Prediction Errors (SSPE) of the model is minimized. This approach is applicable to RPCA and MWPCA, since the Q-statistics are the squared prediction errors of a PCA model. Extending this error minimization approach to the adaptive PCA setting, we may consider the values of η and H that minimize the SSPE with the constraint that the model produces monitoring statistics that are stable enough to fit with good control limits. Similar to this approach, Wold (1994) described a method for selecting the forgetting parameters by first finding the forgetting factors of EWMA charts on the scores of a PCA model to use, and then using them as weights in a weighted formulation of the NIPALS algorithm to identify the PCA loadings.

This approach is similar to the one we will describe, but it attempts to maximize the accuracy of the EWMA models on the scores rather than the accuracy of the PCA model to which they are applied, whereas we are interested in RPCA and MWPCA, which perform monitoring using statistics derived directly from the PCA model instead of based on an ancillary model applied to the scores.

To begin, a range for the candidate values of the parameter δ must be given. MWPCA requires the selection of a window size of H observations. The maximum feasible value of H is C, since we cannot exceed the size of the calibration data set, $\boldsymbol{X}_c$, to build the initial PCA model. A strict lower bound for the window size is that one observation more than the number of retained components are used for modelling. The full range of values which the η parameter in RPCA can take is $0 < \eta \leqslant 1$. However, since RPCA uses exponential downweighting, a value of η rarely needs to be less than 0.9. A choice of η that is too low will generally result in poor monitoring performance. It is possible that the value of the forgetting parameter minimizing the SSPE will forget older observations so quickly that the instability of the model makes it impossible to identify control limits with the desired FDR. Therefore, the range $0.9 \leqslant \eta \leqslant 0.9999$ is commonly taken, though lower values may be necessary for highly non-stationary processes.

To evaluate a candidate value of δ, a model $m_C(\bar{\boldsymbol{x}}_C, \boldsymbol{P}_C, \delta, \boldsymbol{\alpha})$ is specified where $\alpha_{T^2} = \alpha_Q = \varepsilon$, where ε is a small number, such as machine epsilon. This choice of $\boldsymbol{\alpha}$ guarantees that all observations will be used in updating. This is desired since the observations in $\boldsymbol{X}_c$ and $\boldsymbol{X}_v$ come from NOC and in this step, the goal is to explore the performance of the adaptive PCA model for different forgetting parameter values on the validation data set, rather than fault detection. Then, given a validation data set $\boldsymbol{X}_v$, the sum of squared prediction errors of the adaptive PCA model for each observation in $\boldsymbol{X}_v$ is

$$SSPE(\boldsymbol{X}_v) = \sum_{t=C+1}^{V+C} ||\boldsymbol{x}_t - \hat{\boldsymbol{x}}_{t|t-1}||^2 \tag{5.1}$$

where $\hat{\boldsymbol{x}}_{t|t-1}$ is the one-step ahead prediction of $\boldsymbol{x}_t$ given the PCA model from time $t-1$. The $SSPE(\boldsymbol{X}_v)$ can be used to select δ^* in two ways. In the event that there is a distinct minimum, using an exact approach is desirable. This implies the following objective function:

$$\delta^* = \underset{\delta}{\operatorname{argmin}}\ SSPE(\boldsymbol{X}_v). \tag{5.2}$$

A minimization algorithm searching over the feasible range of δ can be used to find δ^*. One such algorithm is the MATLAB `fminbnd` function (which uses a combination of golden section search and parabolic interpolation).

For MWPCA, plotting the SSPE against the candidate window sizes is a good option, with an interpretable rationale. For RPCA, we find it is more informative to plot the SSPE against $2/(1-\eta)$. This formulation has two advantages. First, a grid search in this transformed domain focuses most of the attention on the region of candidate values between 0.99 and 0.9999, which is where we expect many optimal η values to fall in. Secondly, this gives a rough comparison to MWPCA based on the relationship discussed in Section 2.6.4.

5.2.2 Parametrizing control limits

Conventionally, one would set the values of α_{T^2} and α_Q such that they obtain the desired FDR for their respective monitoring statistic, i.e. $P(T^2 > c_{T^2}) = \alpha_{T^2}$ and $P(Q > c_Q) = \alpha_Q$. Moreover, we often have a global FDR (FDR_G) in mind, i.e. $FDR_G = P((T^2 > c_{T^2}) \cup (Q > c_Q)) = \alpha_G$ with α_G typically being 1% or 5%. Hence if we set $\alpha_{T^2} = \alpha_Q = \alpha_G/2$, we would achieve a FDR_G of at most α_G. However, for non-stationary data the control limits (2.2) and (2.3) are not valid and as a consequence FDRs based on these choices of α_{T^2} and α_Q are often higher than desired. This disconnect between the α values and the FDR values can be accounted for by modifying the α values to give the desired FDR.

For non-adaptive PCA methods, it suffices to set the control limits to a constant value giving the desired FDRs for each of the monitoring statistics on $\boldsymbol{X}_v$ since the monitoring model does not change. This will be the approach followed in this paper for the construction of Static PCA control charts. Specifically, we will fit a Static PCA model to a calibration data set and then fix an *empirical* cut-off that gives the desired FDR on a validation data set. However, adaptive methods can vary on the loading matrix ($\boldsymbol{P}$) and number of retained components (k) as the process evolves, resulting in large changes in the control limits. To ensure that the adaptive methods achieve the desired FDR on NOC data, even though the control limits are not constant, we search for values of α_{T^2} and α_Q which, when used as input in the analytical expressions (2.2) and (2.3) for the control

limits of these statistics, result in an $FDR_{T^2} = FDR_Q = FDR_G/2$. If it holds that $P((T^2 > c_{T^2}) \cap (Q > c_Q)) = 0$, we then obtain the desired global FDR. Note that this assumption may not hold perfectly, but if no true faults occur during the NOC calibration and validation data, then the chances that both types of false alarms occur at the same time is low, given that they are random.

To select the so-called *tuned* $\alpha^*_{T^2}$, we first set α_Q to a small value ε, such as machine epsilon, so that no detection occurs on the Q-statistic. For a given time t, we monitor the observation $\boldsymbol{x}_t$ using the model $m_{t-1}(\bar{\boldsymbol{x}}_{t-1}, \boldsymbol{P}_{t-1,k}, \delta^*, \boldsymbol{\alpha})$ with $\boldsymbol{\alpha} = (\alpha_{T^2}, \varepsilon)'$. Then we search for a value of $\alpha_{T^2} \in [\varepsilon, 1-\varepsilon]$ satisfying the following relation:

$$\sum_{t=C+1}^{V+C} I\left(T^2(\boldsymbol{x}_t, \bar{\boldsymbol{x}}_{t-1}, \boldsymbol{P}_{t-1,k}) \geqslant c_{T^2}(k, \alpha_{T^2})\right) = \left\lceil \frac{FDR_G}{2} \times V \right\rceil, \tag{5.3}$$

where $I(.)$ is an indicator function taking value 1 when the argument holds, and 0 otherwise. We thus select $\alpha^*_{T^2}$ such that the FDR on the validation set is $FDR_G/2$. A value of α_{T^2} satisfying Equation (5.3) can be found using a bounded root-finding algorithm (e.g. the MATLAB function `fzero`, which uses a combination of bisection, secant, and inverse quadratic interpolation methods). To select α_Q, we use a similar procedure, but set $\alpha_{T^2} = \varepsilon$ and solve for α^*_Q. Note that the resulting adjusted values of α_{T^2} and α_Q do not correspond to the statistical significance of the monitoring statistics, since the theoretical expressions for the control limits are not valid under the conditions considered in this study. Further, they do not need to be equal to each other.

5.3 Simulations

To illustrate the difference in performance that we obtain by using adjusted values of α_{T^2} and α_Q we show simulation results for a non-stationary process. To obtain each observation at time t we began by generating five scores, $\boldsymbol{y}_t$, for an i.i.d. process according to Equation (5.4):

$$\boldsymbol{y}_t = \boldsymbol{\varepsilon}_t \tag{5.4}$$

with $\boldsymbol{\varepsilon}_t \sim \mathcal{N}(\boldsymbol{0}_5, 0.01\boldsymbol{I}_5)$, where $\boldsymbol{I}_5$ is the 5×5 identity matrix. To make this process non-stationary, we add an increasing and decreasing ramp to the first score after 500 i.i.d. observations. This ramp impacts the mean of the score, increasing it from zero to ten over a period of 500 observations, and then decreasing it to zero again over the next 500 observations followed by another period of 500 i.i.d. observations before the described process repeats. Ramp behavior like this has been used to test adaptive PCA methods before (e.g. Choi

et al. (2006)). These scores are then transformed into a 50-dimensional data set of measurements computed as

$$\boldsymbol{x}_t = \boldsymbol{P}_0 \boldsymbol{y}_t + \boldsymbol{e}_t, \tag{5.5}$$

where $\boldsymbol{P}_0$ is a 50×5 matrix with orthogonal columns randomly generated once and kept constant for all simulation runs. The $\boldsymbol{e}_t$ are 50×1 vectors of white noise errors, distributed as $\mathcal{N}(\boldsymbol{0}_{50}, 0.000025\boldsymbol{I}_{50})$, that simulate measurement noise, as is done, for instance, in Ku et al. (1995) and Lakshminarayanan et al. (1997). The $\boldsymbol{e}_t$ can be seen as the error at the sensor level, and are set to a small value here under the assumption that sensors are typically reliable. Figure 5.1 (left of the mid-line) plots the resulting data. We use the first 2000 observations as calibration data used to fit a wide range of candidate models with different forgetting parameters. The subsequent 2000 observations are used to validate them. By using such a large validation set, we can observe the performance of each candidate model parametrization over a wide range of the potential dynamics it exhibits. In practice, large calibration/validation data sets are ideal, but as a minimum, these data sets should include behavior that is representative of the dynamics of the process. After a model is parametrized, monitoring statistics are computed on the subsequent 4000 observations arising from that process, starting from the stable i.i.d. period. At observations 7100 and 7500 in this test period, large faults are introduced into the process. These are created by taking samples from simulated non-stationary periods and introducing those out of context observations into a stationary period. An example of such a test set can be seen in Figure 5.1 (right of the mid-line).

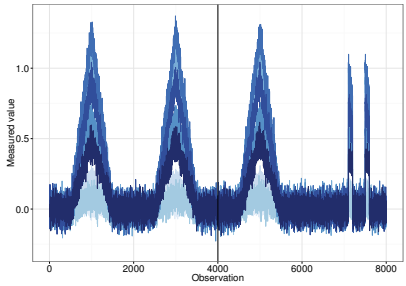

Figure 5.1: A plot of the calibration and validation data (left of the mid-line), and test data with burst faults (right of the mid-line) used for the simulation. The variables from the data are plotted in shades of blue to improve the contrast between them.

We first select the forgetting parameters by consulting the SSPE curves in Figure 5.2 for a range of values for η (left) and H (right). For RPCA, we show

the SSPE curve for values of η between 0.9 and 0.999 to improve interpretation, but larger values of η do not change the qualitative impression of the curve. Similarly, for MWPCA, we show the curve for values of H between the number of variables, p, and the size of $\boldsymbol{X}_c$. The values shown are based on a grid of thirty points and the minimum (if it is distinct from the grid points) for the purpose of visualizing the curve. The minimum is marked by a circle. The value of η minimizing the SSPE is 0.994, and the minimum of the MWPCA SSPE curve is found for $H = 1502$. The forgetting factor η is plotted after being transformed with Equation (2.4). We see that the selected η does not closely correspond to the selected window size for MWPCA, given the relation detailed in Equation (2.4). However, both SSPE curves do display distinct minima. After selecting the forgetting factor, we determine $\alpha^*_{T^2}$ and α^*_Q according to the procedure outlined in Section 5.2.2.

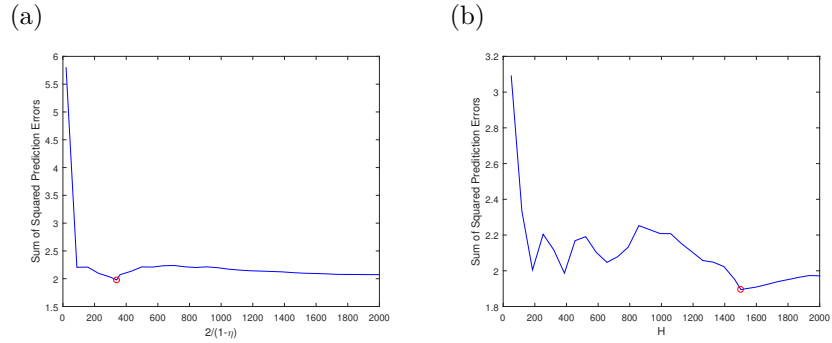

Figure 5.2: (a) Sum of the squared errors of RPCA, and (b) MWPCA for a range of values of η and H, applied to the non-stationary process.

Before examining fault detection, a version of the process without the introduced faults is monitored. The objective of this exercise is to determine how well each of the methods can model the process and obtain the desired false detection rate of 1%. In Table 5.1, we summarize the performance of both adaptive methods using conventional and tuned limits. Conventional limits were imposed by setting $\alpha_{T^2} = \alpha_Q = 0.005$ before online monitoring begins. As a benchmark, we also give results for Static PCA. Static PCA with empirically tuned limits has a global FDR of 1.4%, which is slightly high. We also see in Figure 5.3 that periods corresponding to the troughs have T^2 statistics that are clearly influenced by the non-stationary periods in the process. The base ten logarithm of the T^2 and Q-statistics are used so that large values do not distort the scale, but the monitoring performance is unaffected.

Table 5.1: Detection performance of adaptive PCA control charts on the non-stationary process.

Method	δ^*	Limits	α_{T2}	α_Q	FDR_{T2}	FDR_Q	FDR_G
Static PCA		empirical			0.7%	0.7%	1.4%
RPCA	0.994	conventional	0.005	0.005	0.6%	1.9%	2.5%
RPCA	0.994	tuned	0.005	4.08×10^{-4}	0.5%	0.4%	0.9%
MWPCA	1502	conventional	0.005	0.005	2.62%	1.15%	3.75%
MWPCA	1502	tuned	9.43×10^{-4}	1.24×10^{-3}	0.4%	0.3%	0.7%

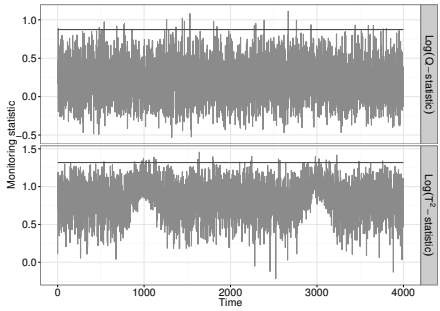

Figure 5.3: Static PCA control charts based on empirical limits applied to the non-stationary process.

Turning to the adaptive methods, we see in Figure 5.4 that RPCA using conventional limits and tuned limits also shows the influence of the non-stationarity. The detection rates based on these different limits give an FDR_G of 2.5% and 0.9% respectively. Of note are the regions of higher values in the Q-statistics. These correspond to the peaks of the non-stationary period, which are the times where the process exhibits the highest complexity. In order to cope with this complexity, the RPCA adapts by increasing the number of components available to itself for modelling. Despite this change in the model, the limits remain valid, and this does not affect the FDR_G. Figure 5.5 illustrates that MWPCA using the conventional limits exhibits a high FDR at the peaks of the non-stationary moments. In contrast, using the tuned limits, MWPCA adapts successfully to the ramps and delivers detection results similar to the RPCA model with tuned limits. Again, the model accounts for the complexity exhibited by the non-stationarity by increasing the number of components, and the distribution of the Q-statistics changes, and on average the monitoring statistics take higher values during the peaks in the simulation.

Figures 5.6 and 5.7 present the control charts corresponding to Static PCA and the adaptive methods using tuned limits when faults are introduced. In

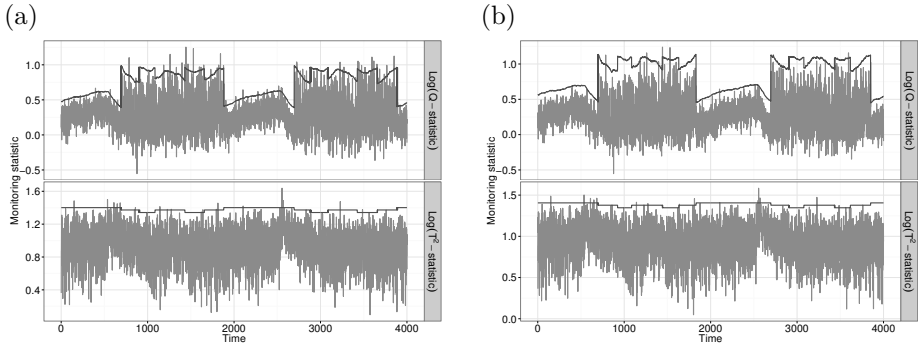

Figure 5.4: RPCA control charts based on (a) conventional and (b) tuned α values applied to the non-stationary process.

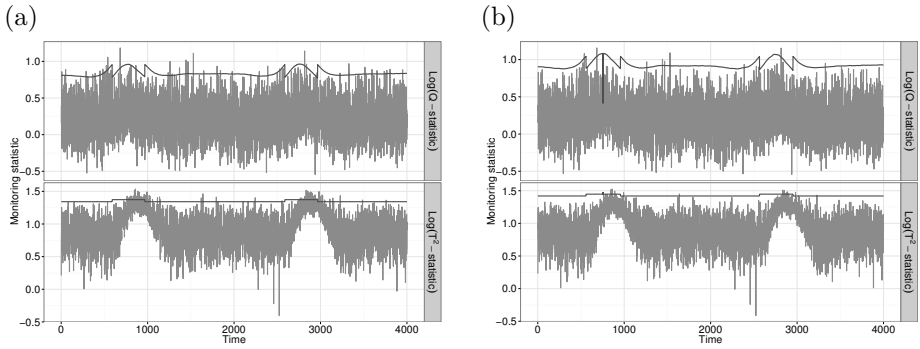

Figure 5.5: MWPCA control charts based on (a) conventional and (b) tuned α values applied to the non-stationary process.

Figure 5.6 a small aberation in the T^2-statistic of the Static PCA control chart is visible after the introduction of the faults, but because the limits have been raised to account for the ramp, the fault is not large enough to detect. Unlike Static PCA, RPCA and MWPCA are able to clearly detect this fault using both monitoring statistics. This is because the methods have adapted to the non-stationary behavior prior to the faults, and in this context they stand out clearly. Since Static PCA was fitted with the ramps the faults are masked to it, whereas if it was not fitted to the ramps, the natural ramps would have resulted in false alarms. A well known danger of adaptive methods is that they can eventually adapt to faults, but as this example shows, applying non-adaptive methods to non-stationary processes can mean forfeiting the chance to detect certain types of obvious faults.

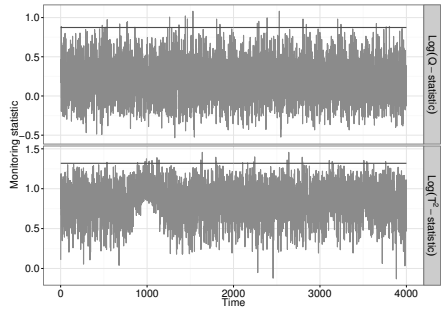

Figure 5.6: Static PCA control charts when the test data contain faults.

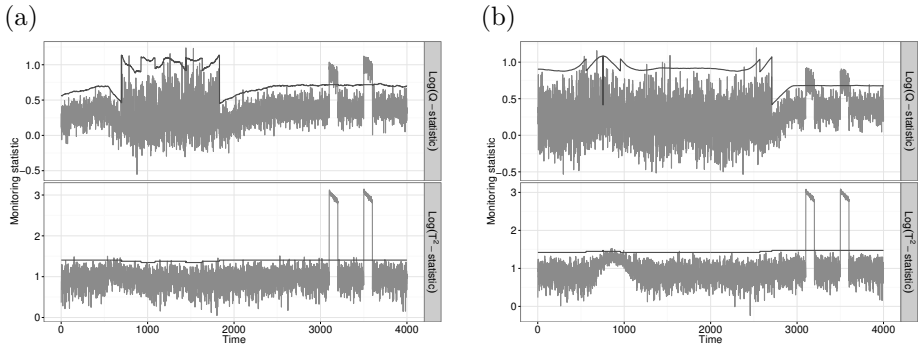

Figure 5.7: (a) RPCA and (b) MWPCA control charts when the test data contain faults.

5.4 Case studies

For the simulation, the adaptive models using tuned limits achieved FDR values close to the desired 1% level. Furthermore, we also saw that the MWPCA model using conventional limits led to FDR values considerably higher than 1%. Since these results were obtained on NOC data, the parameter selection based on the calibration and validation data sets are confirmed to be appropriate and effectively reduce the number of false alarms. In the following case studies the capability of the proposed procedure to also provide improved fault detection will be illustrated in two real-life industrial processes.

5.4.1 The NASA process

As basis for comparison with the adaptive methods, we fit a Static PCA model to the NASA process, introduced in Chapter 2, Section 2.2. The first 300 observations of the series were used to fit the PCA model and respective empirical control limits for the monitoring statistics. These control limits are chosen such that both monitoring statistics attain a 0.5% FDR on the validation data set, leading to an expected global FDR of 1%. The calibration and validation data sets (denoted as before as $\boldsymbol{X}_c$ and $\boldsymbol{X}_v$) were 150 observations each. The monitoring results for the remaining observations are shown in Figure 5.8. The Static PCA control charts for the NASA process detect the onset of failure at around $t = 700$; much earlier than the catastrophic failure of the entire system. This detection reflects the visible increase in the vibrations of the sensors attached to the fourth bearing. Although both monitoring statistics signal many faults, the T^2-statistic is particularly sensitive. The global Detection Rate (DR_G) of this Static PCA model is 79%, meaning that 79% of observations are identified as a fault by at least one of the monitoring statistics. Early detection was expected of Static PCA, and will serve as a point of comparison to the adaptive methods, which will tend to signal the fault later. The desirability of earlier or later detection here depends on the actual impact of the vibration on process performance.

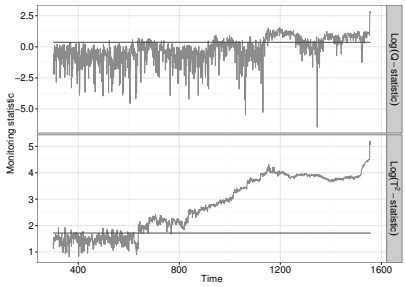

Figure 5.8: Static PCA control charts for the NASA process.

To analyze the NASA process with adaptive methods, we first select forgetting parameters for RPCA and MWPCA. We use the same calibration/validation scheme as we used to specify the Static PCA model. In Figure 5.9 we show the SSPE curves over a range of values for η (left) and H (right). The value of η minimizing the SSPE for RPCA is 0.976. For MWPCA, the SSPE curve exhibits a long tail. However, the minimum is at $H = 58$, which is also close to the elbow, so we use this value. We note that the range covered by these two curves is

different. Partially, this is because RPCA can complete models corresponding to larger forgetting parameters in the validation data set than the calibration data set permits MWPCA to use. In the simulations, we circumvented this intrinsic difference of the methods by using large calibration data sets that guaranteed that models accurately reflected their forgetting parameters before validation was performed, but in applications this is not always possible. However, here we see that the selected forgetting parameters are both minima occurring at low values in the candidate range.

Table 5.2 and Figure 5.10 display the parameter selection and monitoring results for the PCA methods. First, considering the RPCA model, the conventional limits result in detection of the process fault at around $t = 900$ (Figure 5.10(a)). At this time enough observations are out-of-control (either statistic exceeds its control limit) with respect to the model, which no longer updates, and thus most of subsequent observations are declared faults by both monitoring statistics ($DR = 56\%$). In contrast, the RPCA model with tuned control limits continues to adapt until the beginning of the catastrophic failure. These plots perform differently because the conventional limits are lower than the tuned limits, resulting in more conservative monitoring performance. Depending on the preference of the practitioner, in this case, the conventional limits may actually lead to a preferable detection time. However, this does not mean that these limits are appropriate for this model. Rather, the forgetting factor selected for RPCA leads to adaptation, and the tuned limits are more aligned with this parameter choice, and give an FDR closer to the desired one. If the goal is early detection, then Static PCA is a more suitable choice since the process is intrinsically stationary.

Table 5.2: Detection performance of PCA control charts on the NASA process.

Method	δ^*	Limits	α_{T^2}	α_Q	DR_{T^2}	DR_Q	DR_g
Static PCA		empirical			55%	76%	79%
RPCA	0.976	conventional	0.005	0.005	54%	45%	56%
RPCA	0.976	tuned	5.09×10^{-7}	1.75×10^{-4}	3%	4%	4%
MWPCA	58	conventional	0.005	0.005	74.1%	60%	76.1%
MWPCA	58	tuned	5.12×10^{-7}	2.10×10^{-4}	3.5%	4.5%	5%

Figure 5.11 contains analogous control charts based on MWPCA. The control charts based on conventional limits and those of Static PCA are similar. Since the MWPCA control chart using conventional limits detects the faults early, like Static PCA, which is not adaptive, this is an indication that the α values used to construct this chart may by too low for an adaptive model. The control chart based on tuned α values shows later detection, and these monitoring statistics are nearly identical to those of RPCA based on tuned α values. On closer

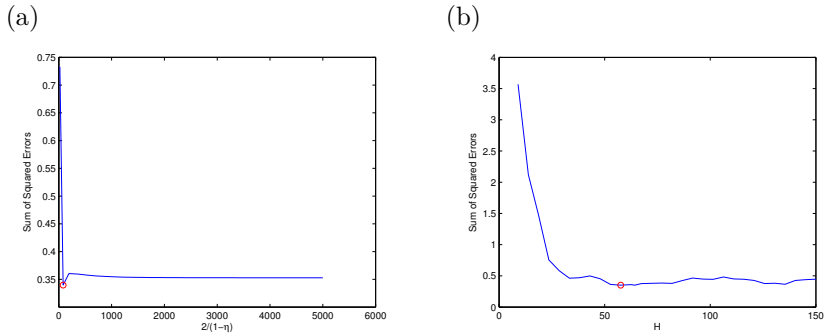

Figure 5.9: Sum of the squared prediction errors of (a) RPCA and (b) MWPCA for a range of values of η and H, applied to the NASA process.

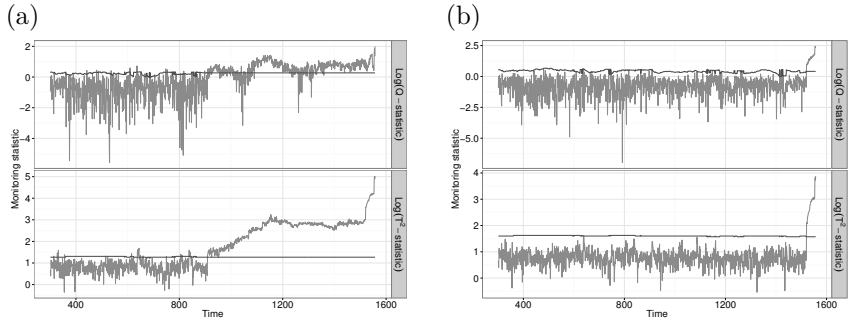

Figure 5.10: RPCA control charts using (a) conventional and (b) tuned α values for the NASA process. Both versions use $\eta = 0.976$.

inspection, we also see that the tuned α parameters of the MWPCA are quite similar to those of RPCA, meaning these models may be nearly equivalent.

Note that this low-dimensional data set could also be monitored by modeling the time series, and evaluating the resulting residuals. Such a parametric approach could give rise to FDRs closer to the desired level than the nonparametric PCA procedures studied here, but it also requires a careful choice and estimating procedure of the time-varying model. We refer to De Ketelaere et al. (2016) for some references and a novel approach based on co-integration.

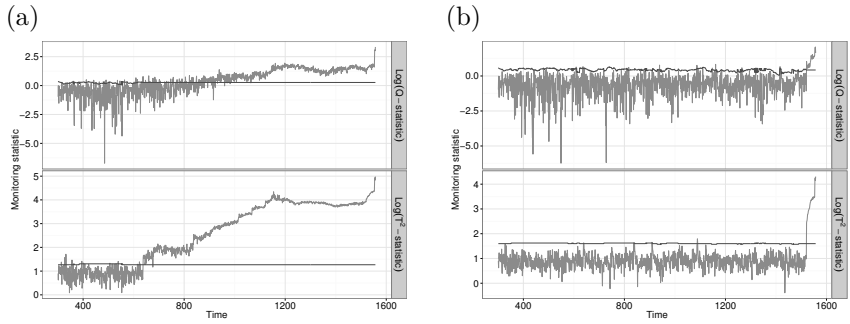

Figure 5.11: MWPCA control charts using (a) conventional and (b) tuned α values for the NASA process. Both versions use $H = 58$.

5.4.2 The Stamping process

The second real data example we consider comes from an industrial stamping process De Ketelaere et al. (2011). Each observation corresponds to 123 vibration measurements (in voltage) over the course of a single stamping event. In total, 11519 stamping events were monitored; one each second. Since the data is high-dimensional, PCA is one of the few monitoring approaches that can be feasibly used. In Figure 5.12(a), we show a sample of 500 stamping events. A typical stamping event displays an identifiable vibration pattern. The goal of an SPM procedure is to detect events which do not conform to this pattern. In this data set, faults are observations which display atypically low, high, or asynchronous vibrations relative to the majority of events. The events with very low vibrations at around $t = 80$, or the events with high vibration at $t = 90$ are examples. Another feature of the stamping process is that it naturally exhibits a mild non-stationarity. In Figure 5.12(b), we show the maximum vibration intensities from 3000 stamping events to give a general sense of the behavior of the process over time. Ignoring the faults, we can see that there is a regular undulation in this process that reflects normal behavior and should not be flagged as faulty. Adaptive SPM methods are appropriate for this reason. Furthermore, although adaptive methods can adapt to persistent faults, since the faults in this process are single events, they do not pose this problem.

Figure 5.13 shows Static PCA control charts for the Stamping process. We used the first 1400 observations from this data set to fit and validate the model. Since faults are present throughout this data set, it is difficult to isolate a clean period in which to fit a model. Hence, we first performed ROBPCA [Hubert et al. (2005)], a robust PCA procedure, to remove faults during this time period.

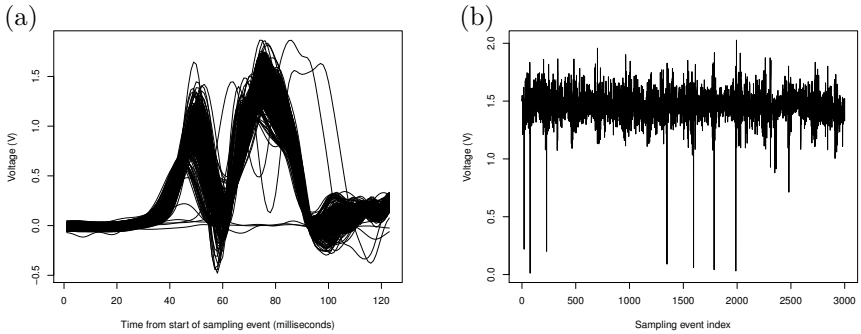

Figure 5.12: (a) Sample of 500 events from the Stamping data; (b) Maximum vibration intensities from 3000 sample events.

Since there are not so many faults in this period, the parameter controlling the robustness of the algorithm was set to exclude only large outliers. This resulted in 1299 NOC observations to train and validate the models with. This was necessary because faults present in the calibration and validation data ($\boldsymbol{X}_c$ and $\boldsymbol{X}_v$) influence the fitted control limits, increasing the chances that faults are considered normal. $\boldsymbol{X}_c$ and $\boldsymbol{X}_v$ were 650 and 649 observations each. A robust PCA procedure was chosen since it is a PCA method used to identify atypical observations, like the adaptive PCA control charts. However, since ROBPCA is a Static PCA technique, and does not account for process dynamics, the outlier detection rate of 7% may be higher than the true fault rate, potentially leading to overly conservative control limits. Although this is not the most ideal solution, we are not aware of a robust PCA option that does not follow a Static PCA approach, and including major faults in the calibration and validation data is even more detrimental. After fitting the Static PCA model on the cleaned data we obtained monitoring statistics for the next 2000 observations. Actual faults occur throughout the process in clusters, rather than as a single, large failure. Static PCA is not well-suited for monitoring this type of process since it assumes stationarity, while the actual data exhibits mild non-stationarity. We see that its inability to account for the non-stationarity in this process results in many observations, particularly those at points in the process cycle with natural high vibrations, being flagged as faults. The resulting DR is 11%.

In Figure 5.14 we show the SSPE curves over a range of values for η (left) and H (right) for the Stamping data. The value of η minimizing the SSPE for RPCA is 0.999. For MWPCA the minimum is at $H = 254$.

With models using these forgetting parameters, we next select values for the α

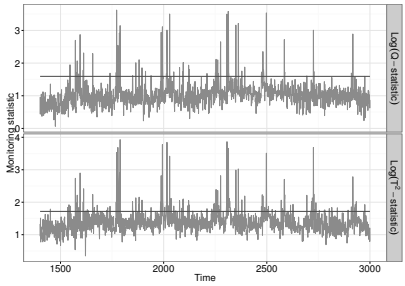

Figure 5.13: Static PCA control charts for the Stamping process.

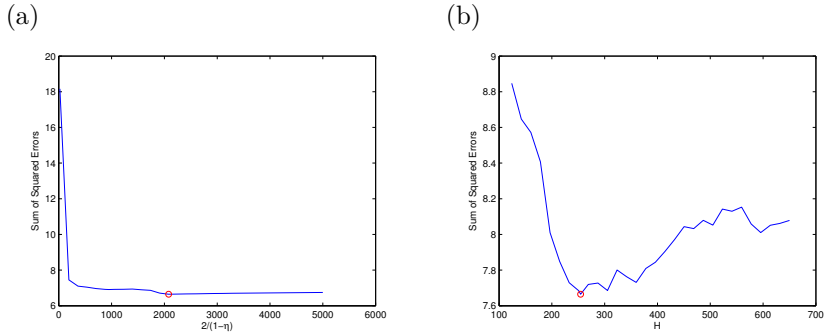

Figure 5.14: Sum of the squared prediction errors of (a) RPCA and (b) MWPCA for a range of values of η and H, applied to the Stamping process.

parameters and produce control charts. Consulting Figure 5.15 and Table 5.3, RPCA based on conventional limits shows many more alarms ($DR = 21\%$) than Static PCA using empirical control limits. However, the detection rate of the tuned model ($DR = 6\%$) is more adequate since this model allows many of the in-control observations to pass, while conventional RPCA and Static PCA considered them as faults. Instead, alarms are primarily signaled by large, relevant faults. Again, this performance improvement is accomplished because the tuned α values are much lower than the conventional values, resulting in higher and more accurate control limits.

MWPCA performs similarly to RPCA (Figure 5.16). The conventional control limits lead to a high detection rate. On the other hand, the tuned limits have a $DR = 4\%$, which is close to the $DR = 6\%$ of the tuned RPCA model. Again, this lower detection rate seems more accurate than that of Static PCA and

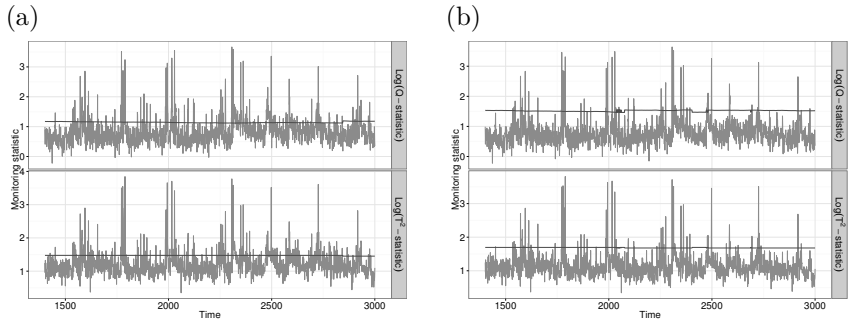

Figure 5.15: RPCA control charts using (a) conventional and (b) tuned α values for the Stamping process. Both versions use $\eta = 0.999$.

Table 5.3: Detection performance of PCA control charts on the Stamping process.

Method	δ^*	Limits	α_{T^2}	α_Q	DR_{T^2}	DR_Q	DR_g
Static PCA		empirical			11%	5%	11%
RPCA	0.999	conventional	0.005	0.005	17%	15%	21%
RPCA	0.999	tuned	3.52×10^{-6}	2.08×10^{-8}	5%	4%	6%
MWPCA	254	conventional	0.005	0.005	29%	35%	39%
MWPCA	254	tuned	7.94×10^{-9}	2.65×10^{-12}	4%	3%	4%

MWPCA using conventional limits because the process is non-stationary and faults are not exceedingly common.

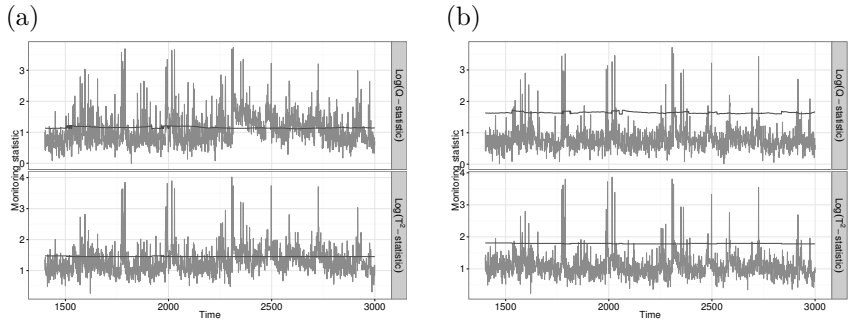

Figure 5.16: MWPCA control charts using (a) conventional and (b) tuned α values for the Stamping process. Both versions use $H = 254$.

Most of the events in this data set are not faults, so we expect a relatively low detection rate. For this reason, it is likely that the difference between the adaptive methods and Static PCA is largely driven by false detections from the Static PCA model. The difference between the detection performances of the adaptive methods and Static PCA is between 5% and 7%. Although this is not an enormous magnitude, in practice, it represents a large improvement in accuracy since reducing the unnecessary rejection of $5\% - 7\%$ of output translates to large savings in a mass-production setting.

To further validate the accuracy of the fault detection of these methods, in Figure 5.17 we plot the curves from the test data with observations classified by the methods as in-control (left) and out-of-control (right). Across the methods, curves classified as in-control follow a recognizable pattern. The curves classified as out-of-control consist of many clear faults, and curves that may also be misclassified as faulty. The main reason for this may be that the data is not stationary. One way that the adaptiveness of RPCA and MWPCA manifests is in the horizontal range of the in-control curves which is wider. In some sense, adaptation here has achieved an effect similar to warping because the transition between the positions of the curves was continuous in time. As a consequence, RPCA and MWPCA flag fewer of normal looking curves as faults. RPCA detects 50% fewer faults than Static PCA without misclassification of any of the obvious faults; for MWPCA, this difference is 63%. In the plots of the out-of-control curves identified by the adaptive methods, this corresponds to thinner congregations of normal looking curves. With no obvious faults classified as in-control by any of the methods, it is plausible that the adaptive methods have achieve a lower false detection rate than Static PCA on this data set.

An alternative way of summarizing the raw data is to use heat maps to characterize each observation visually. In Figure 5.18, heat maps for each monitoring method are displayed with the observations classified as in-control on the top of the heat maps above the black line. The observations classified as out-of-control are below the black line. The contrast between these two classes of observations is pronounced for all of the monitoring methods. In the case of MWPCA and RPCA, there are groups of observations that look somewhat atypical that appear in clusters in time, but this is also the case for PCA, and evidently these clusters are not distinct enough that they are visible in Figure 5.17. The most noticeable difference between the heat maps for the different methods is that many of the out-of-control observations in the PCA heat map are consistent to each other. These are observations that the adaptive methods can adjust to include as in-control, while they lie outside of the behavior that the initial PCA model is fitted on.

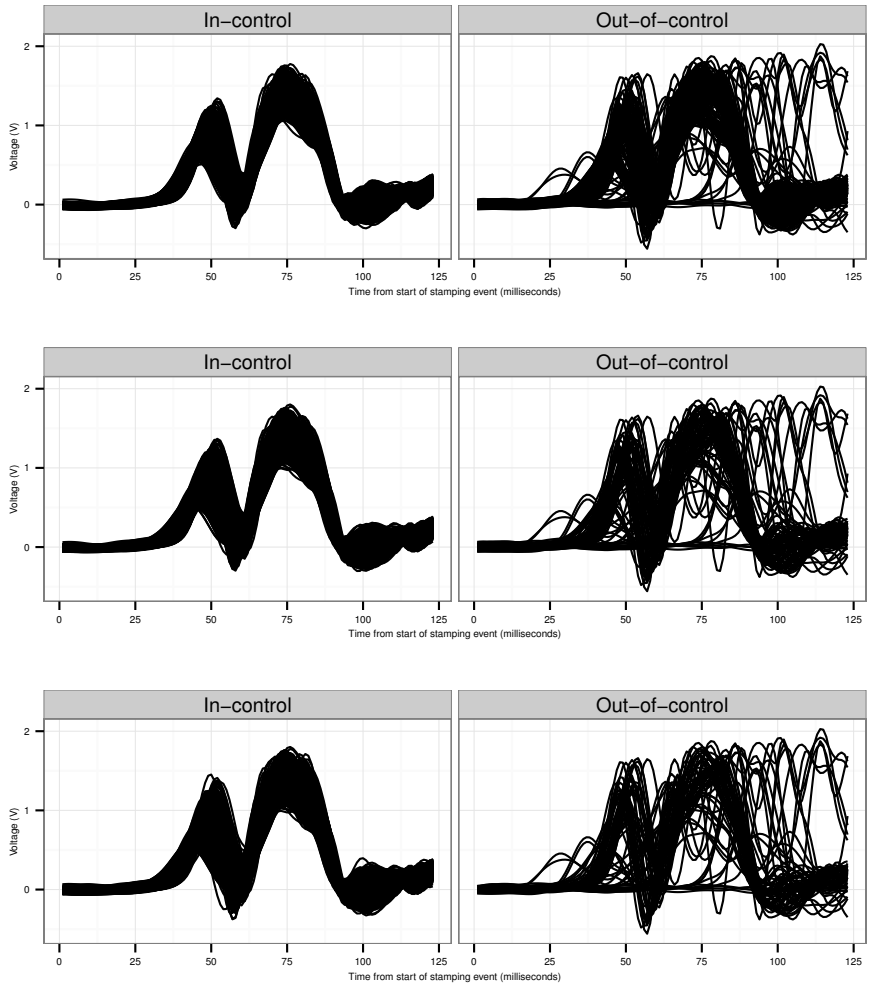

Figure 5.17: Classification of stamping events by static PCA (top), RPCA (middle) and MWPCA (bottom). Observations classified as in-control (left) and out-of-control (right).

5.5 Discussion

Adaptive monitoring methods are useful tools for monitoring processes with natural non-stationarity, but their use also introduces the potential of adapting

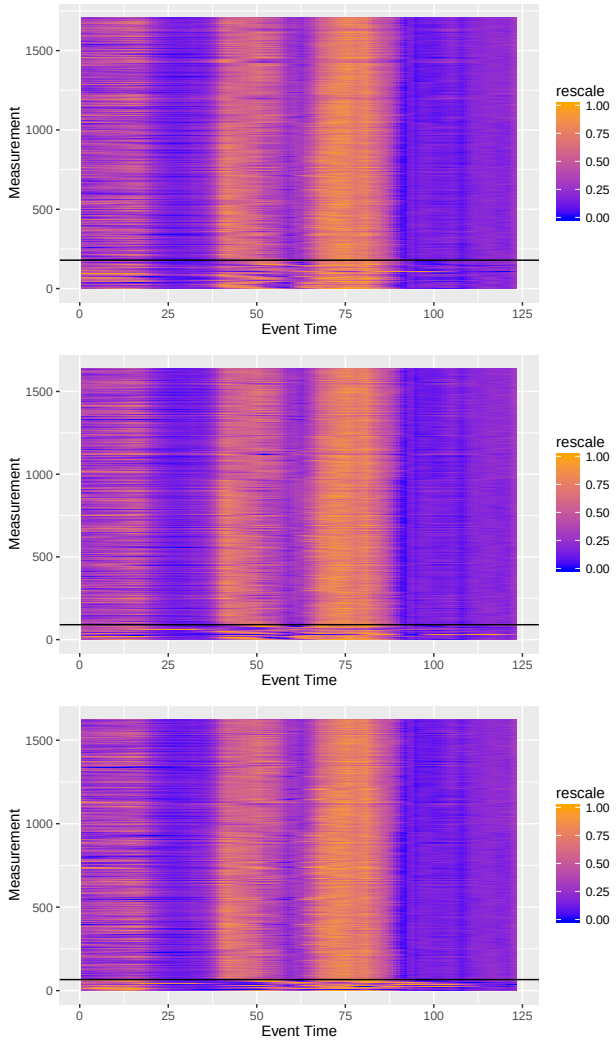

Figure 5.18: Heat maps of stamping events classified by static PCA (top), RPCA (middle) and MWPCA (bottom). The black lines separate in (top) and out (bottom) of control observations.

to faulty behavior. This remains a real concern for adaptive PCA methods, but given that static methods are often unsuitable for monitoring non-stationary processes, adaptive methods may still be preferable. However, proper application

of these methods relies on their correct parametrization. As shown in Section 5.3, the more a process violates the assumptions of the model monitoring it, the more carefully parameters, such as the α values determining the control limits, need to be selected. The NASA data example is also an illustration of the balanced parameter selection required by adaptive methods. Here, though the process is known to be stationary, it can be used to explore how adaptive methods can be tuned to react to non-stationarity of different levels. In this example, we found that MWPCA and RPCA could be automatically tuned to give (depending on the objectives of the operator) good fault identification behavior in the sense that they adapt to potentially ignorable non-stationarity, but still detect the catastrophic failure in advance of its occurrence. We also applied Static PCA and the adaptive methods to the naturally non-stationary Stamping process and found that the most realistic performance was given by the adaptive methods using tuned parameters.

5.6 Conclusions

In this chapter, we have provided guidelines for the selection of the forgetting parameter in RPCA and MWPCA, and an approach for selecting the α parameters that give a detection performance in line with actual expectations. We find that our approach of selecting the forgetting parameter based on the SSPE generally leads to effective models. Similarly we find that by tuning the value of the α parameters to account for the imperfections of the monitoring models leads to more realistic results. On this point, we believe that the fact that adjusting the α parameters is necessary, indicates that improvements in the modelling methods are also needed. Indeed, as the model adapts, the tuned α values can become less accurate, and should ideally be re-tuned. This online re-tuning is rarely possible in practice though, and the best protection against issues this might cause is therefore an accurate model. There is however, a trade off between accuracy and ease of implementation and between general methods (such as PCA-based methods) and more process specific methods (e.g. physical models). Furthermore, even improved methods may still require adjustments to their control limits since the complexity of modelling non-stationary processes means that even a flexible, accurate approach is likely to achieve less than perfect performance.

Chapter 6

Sparse PCA for high-dimensional data with outliers

Based on: Hubert, M., Reynkens, T., Schmitt, E., Verdonck, T. (2015). Sparse PCA for high-dimensional data with outliers. *Technometrics. Accepted.* DOI: 10.1080/00401706.2015.1093962. Tom Reynkens and Eric Schmitt contributed equally to this paper.

6.1 Introduction

Principal Component Analysis (PCA) is a popular technique used for dimension reduction. The idea is to find a number of uncorrelated linear combinations of the original variables that capture most of the covariance structure of the original data. These combinations are called the Principal Components (PCs). Those directions are chosen such that they are orthogonal and sequentially maximize the variance of the projected data. Typically one does not use all the PCs, but only the first k explaining a sufficient portion of the total variance (i.e. information) of the original data. Despite its advantages, Classical Principal Component Analysis (CPCA) also has several drawbacks(CPCA is another name for Static PCA that is more conventional in the robust PCA literature. It will be used in this chapter); two of which we will focus on.

First, CPCA often results in PCs that are difficult to interpret because most of the loadings are neither very small nor very large in absolute value. To increase interpretability, sparse PCA methods were developed to estimate PCs with many zero loadings. This is useful when the data is high-dimensional, since only a subset of the original variables may need to be analyzed or measured. Two popular methods for performing sparse PCA are SCoTLASS [Jolliffe et al. (2003)] and SPCA [Zou et al. (2006)].

Second, it is well known that outliers present in the data can heavily effect the CPCA estimates. Several robust alternatives for CPCA have been proposed including a Projection Pursuit PCA approach (PP-PCA) [Li and Chen (1985); Hubert et al. (2002); Croux and Ruiz-Gazen (2005)], spherical PCA [Locantore et al. (1999)], and ROBPCA [Hubert et al. (2005)].

In this paper, we propose a new method, RObust Sparse PCA (ROSPCA), combining the advantageous properties of sparse and robust PCA. Previous work on this problem has been done by Croux et al. (2013), who developed a sparse version of the robust PP-PCA method by integrating sparsity principles into the formulation of PP-PCA. Since we believe that the detection of outliers may be the more difficult, and crucial, challenge, we approach the problem from a different direction, and develop a sparse modification of the robust ROBPCA method. The main difference is that we partially separate the outlier detection step from the sparsification step. As we detail in this paper, doing so results in greater robustness and more accurate sparse estimates.

Note that our model assumptions are different from those studied in Candès et al. (2011) and Zhou et al. (2010). Whereas we are searching for a subspace spanned by sparse vectors, in the latter papers not the subspace but the errors are supposed to be sparse. This allows to recover the subspace exactly with a convex optimization program.

In Section 6.2 we first give a summary of existing methods for sparse and/or robust PCA, and then we detail our new method together with a new criterion to select the sparsity parameter. Section 6.3 contains the results of a simulation study, whereas Section 6.4 illustrates ROSPCA on a real dataset. Finally, Section 6.6 contains conclusions and directions for further research.

6.2 Methods

6.2.1 Classical PCA

To fix notation, we begin by defining PCA for a data matrix, $\boldsymbol{X} = \boldsymbol{X}_{n,p} \in \mathbb{R}^{n \times p}$. In general, the subscripts denote the dimensions of the matrix and will only be added when appropriate. The p-dimensional observations in $\boldsymbol{X}$ are denoted by $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$. The loadings of the PCs, i.e. the components of the linear combinations, are in the columns of the orthogonal *loadings matrix* $\boldsymbol{P}$. Given estimated loadings $\boldsymbol{P}$ and center $\hat{\boldsymbol{\mu}}$, projecting the centered $\boldsymbol{X}$ on the new directions yields the *scores matrix* $\boldsymbol{T} = (\boldsymbol{X} - \mathbf{1}_n \hat{\boldsymbol{\mu}}')\boldsymbol{P}$, with $\mathbf{1}_n$ a column vector consisting of n ones.

Classical PCA can be described as searching for a $\hat{\boldsymbol{\mu}}$ and $\boldsymbol{P}$ such that the scores have maximal variance, and are uncorrelated. The PCA directions then correspond to the eigenvectors of the classical covariance matrix $\boldsymbol{S}$ of $\boldsymbol{X}$, whereas the variance of the data projected on an eigenvector is equal to the corresponding eigenvalue of $\boldsymbol{S}$. Note that when the variances of the original variables differ greatly, the data should first be standardized. If one uses the componentwise standard deviation, this comes down to computing the eigenvectors of the correlation matrix of $\boldsymbol{X}$.

Typically, $k \ll p$ dimensions are needed to express the information in the data. Various approaches exist to select the number of components to retain, k. One of the simplest and most popular is the *scree plot*. It plots the sorted, decreasing eigenvalues versus their index. The number of components corresponding to the point at which an elbow in the plot occurs is then selected. Following the selection of the number of components, only the first k columns of $\boldsymbol{P}$ are used and denoted as $\boldsymbol{P}_{p,k} = [\boldsymbol{p}_1, \ldots, \boldsymbol{p}_k]$.

6.2.2 Sparse PCA

Sparse PCA has the advantage of making the interpretation of the PCs easier. A simple way to accomplish this is to set all loadings with absolute value smaller than a certain threshold to zero. This method is called *simple thresholding*. Cadima and Jolliffe (1995) noticed that this method can be potentially misleading. For example, one should also look at the standard deviations of variables to determine the contribution of a variable to a certain PC.

To overcome the issues of that early method, a number of methods have been developed. One of these is SCoTLASS, which was proposed by Jolliffe et al. (2003). It integrates an L_1 constraint with PCA, yielding sparse loadings. The resulting objective function seeks the orthogonal loadings $\boldsymbol{p}_j$ maximizing the variance explained by the fitted model, subject to the constraint $\|\boldsymbol{p}_j\|_1 \leqslant \eta_j$, a sparsity constraint, where $\|\boldsymbol{p}_j\|_1$ is the L_1 norm of $\boldsymbol{p}_j$. We will work with the dual of this problem:

$$\boldsymbol{p}_j = \underset{\|\boldsymbol{p}\|=1, \boldsymbol{p}\perp\boldsymbol{p}_1,\ldots,\boldsymbol{p}\perp\boldsymbol{p}_{j-1}}{\operatorname{argmax}} \boldsymbol{p}'\boldsymbol{S}\boldsymbol{p} - \lambda_j\|\boldsymbol{p}\|_1, \tag{6.1}$$

where $\boldsymbol{p}_j$ is the jth PCA direction. Under this formulation, λ_j is the sparsity parameter for SCoTLASS, in place of η_j. A higher value of λ_j corresponds to greater sparsity, and a value of zero corresponds to no sparsity.

6.2.3 Robust PCA

The loadings matrix estimated by CPCA and sparse PCA is very sensitive to outliers. Robust principal component analysis addresses this issue. Two well known robust PCA methods are robust Projection Pursuit PCA (PP-PCA) and ROBPCA. PP-PCA maximizes a robust measure of spread to obtain consecutive directions on which the data is projected. Croux and Ruiz-Gazen (2005) proposed a version that serves as the basis for one variant of sparse, robust PCA. The ROBPCA method [Hubert et al. (2005)] combines ideas from projection pursuit and robust covariance estimation. These approaches will be discussed in greater detail below, when we encounter sparse versions.

To detect PCA outliers, two notions of distance are used: robust score distances and orthogonal distances. The *robust score distance* (SD) measures the robust statistical distance from a PC score to the center of the scores. For an observation $\boldsymbol{x}_i$, the robust score distance is defined as

$$\mathrm{SD}_i = \sqrt{\sum_{j=1}^{k} \frac{(\boldsymbol{t}_i)_j^2}{l_j}} = \sqrt{\boldsymbol{t}_i'\boldsymbol{L}^{-1}\boldsymbol{t}_i}, \tag{6.2}$$

with k the number of PCs, $(\boldsymbol{t}_i)_j$ the jth component of the ith score $\boldsymbol{t}_i$ and $\boldsymbol{L}$ the diagonal matrix containing the robust eigenvalues corresponding to the robust PCs. We set the cut-off for observations with high SD values at $c_{\mathrm{SD}} = \sqrt{\chi^2_{k,0.975}}$, the square root of the 97.5% quantile of a chi-squared distribution with k degrees of freedom. This is justified when the scores are approximately normally distributed.

The *orthogonal distance* (OD) of an observation $\boldsymbol{x}_i$ to the PCA subspace is given by

$$\text{OD}_i = \|\boldsymbol{x}_i - \hat{\boldsymbol{\mu}} - \boldsymbol{P}_{p,k}\boldsymbol{t}_i\|. \tag{6.3}$$

Note that $\hat{\boldsymbol{\mu}} + \boldsymbol{P}_{p,k}\boldsymbol{t}_i$ is the projection of $\boldsymbol{x}_i$ on the PCA subspace determined by $\boldsymbol{P}_{p,k}$ and $\hat{\boldsymbol{\mu}}$. To obtain a cut-off for the orthogonal distances, we follow the approach taken in Hubert et al. (2005). This makes use of the Wilson-Hilferty approximation for a chi-squared distribution, which implies that the orthogonal distances to the power 2/3 are approximately normally distributed. To obtain estimates of the center and scale of this distribution we use the univariate MCD [Rousseeuw (1984)], a robust estimator that searches for the subset of size $\frac{n}{2} < h \leqslant n$ that has the smallest variance and bases location ($\hat{\mu}_{MCD}$) and scale ($\hat{\sigma}_{MCD}$) estimates on it. Given these parameters, the cut-off is defined as $c_{\text{OD}} = (\hat{\mu}_{\text{MCD}} + \hat{\sigma}_{\text{MCD}} z_{0.975})^{3/2}$, with $z_{0.975}$ the 97.5% quantile of the standard normal distribution.

6.2.4 SRPCA

Croux et al. (2013) proposed a robust, sparse method that combines ideas from the PP approach and sparse PCA. It will be used as a benchmark in our simulations and a real data example. Their approach consists of adding the L_1 penalty into the PP equations. The method thus looks for directions that maximize the scale of the data projected on them under the constraint that the loadings of these directions should not be too large. The jth sparse PCA direction is given by

$$\tilde{\boldsymbol{p}}_j = \begin{cases} \underset{\|\boldsymbol{p}\|=1}{\text{argmax}} \; S(\boldsymbol{p}'\boldsymbol{x}_1, \ldots, \boldsymbol{p}'\boldsymbol{x}_n) - \lambda_1 \|\boldsymbol{p}\|_1 & \text{if } j = 1 \\ \underset{\|\boldsymbol{p}\|=1,\, \boldsymbol{p} \perp \tilde{\boldsymbol{p}}_1, \ldots, \boldsymbol{p} \perp \tilde{\boldsymbol{p}}_{j-1}}{\text{argmax}} \; S(\boldsymbol{p}'\boldsymbol{x}_1, \ldots, \boldsymbol{p}'\boldsymbol{x}_n) - \lambda_j \|\boldsymbol{p}\|_1 & \text{if } 1 < j \leqslant p, \end{cases} \tag{6.4}$$

where S is a measure of scale. If one uses the sample standard deviation for S, this method is nothing more than SCoTLASS. To obtain robust principal components, Croux et al. (2013) suggest to use the robust Q_n estimator of scale [Rousseeuw and Croux (1993)]. The Q_n is the first quartile of the pairwise distances between the elements of a vector. The data are typically centered using a robust estimator for the center (e.g. using the L_1-median). Then, one applies the PP steps on the $\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}$ (for $1 \leqslant i \leqslant n$), with $\hat{\boldsymbol{\mu}}$ the robust estimate for the center.

The sparsity parameter λ_j can vary across the different PCs. Croux et al. (2013) make the relative importance of the L_1 penalty comparable across the different PCs. This means that there is a similar degree of sparsity across the PCs. They

take $\lambda_j = \lambda v_j$ where v_j can be defined as follows. Suppose we have found the $j-1$ first PC directions and denote by $X_j^{\perp}$ the data projected on the space orthogonal to the space spanned by the $j-1$ first PC directions. The number v_j is then the average of the variance measure S^2 applied to the columns of $X_j^{\perp}$. Note that v_1 is the average of the variance measure S^2 applied to the columns of X. This definition is used in the `R` packages *pcaPP* [Filzmoser et al. (2014)] and *rrcovHD* [Todorov (2014)] and differs slightly from the definition in Croux et al. (2013). Hence, there is only one tuning parameter to select: the sparsity parameter λ. We denote this method by SRPCA as in Todorov and Filzmoser (2013).

To find the sparse PCA directions in (6.4), the expressions need to be maximized over a p-dimensional space. This optimization problem is non-convex. The Grid algorithm of Croux et al. (2007) is an accurate algorithm that is used to obtain the PCA directions in the PP approach. In Croux et al. (2013), the authors extend it for sparse PCA and provide a detailed description of the algorithm. Since SRPCA is a generalization of the PP approach, Croux et al. (2013) proposed to extend the Grid algorithm to compute the sparse directions. Henceforth, we will use this algorithm to compute the sparse loadings of SCoTLASS and SRPCA. By default, the maximum number of iterations is equal to 10, but we noticed that the algorithm does not yet converge then. We use a maximum of 75 iterations instead which provides stable results.

6.2.5 ROSPCA

Hubert et al. (2005) proposed a robust PCA algorithm combining ideas from projection pursuit and the MCD estimator, which they called ROBPCA. Many steps in ROBPCA anticipate those of ROSPCA as the robustness properties of the latter derive almost directly from the former. Intuitively, they can be compared as follows. ROBPCA finds an outlier-free subset which determines a robust subspace. Then, it projects the data onto this subspace to estimate the eigenvectors and eigenvalues robustly. The ROSPCA method (RObust Sparse PCA) integrates sparse PCA into ROBPCA. In doing so, ROSPCA finds a subset that determines a robust, *sparse* subspace, and then estimates the eigenvectors and eigenvalues while preserving sparsity.

Not surprisingly the method contains two hyperparameters: α which determines the degree of robustness and λ which regulates the sparsity. The value of α must satisfy $0.5 \leqslant \alpha < 1$ and needs to be chosen in advance. It constitutes a lower bound on the number of regular observations, so at most $(1-\alpha)100\%$ of the n data points are allowed to be outlying. If no a priori information about the amount of outliers is available, we recommend to set $\alpha = 0.5$, yielding

maximal robustness. The choice of the sparsity parameter λ will be discussed in Section 6.2.6.

The ROSPCA algorithm consists of an outlier detection part (step 1), and a sparsification part (steps 2 and 3):

1. The first part is similar to ROBPCA, so we describe it only shortly. When a standardization is appropriate, the variables are first robustly standardized by means of the componentwise median and the Q_n. Then using the SVD of the resulting data matrix, the p-dimensional data space is reduced to the affine subspace spanned by the n observations. We denote the resulting data matrix (of rank at most $n-1$) by $\tilde{\boldsymbol{X}}$. Next, for each $\tilde{\boldsymbol{x}}_i$ the Stahel-Donoho outlyingness is computed as

$$\text{outl}(\tilde{\boldsymbol{x}}_i) = \max_{\boldsymbol{v} \in B} \frac{|\tilde{\boldsymbol{x}}_i' \boldsymbol{v} - \hat{\mu}_{\text{MCD}}(\tilde{\boldsymbol{x}}_j' \boldsymbol{v})|}{\hat{\sigma}_{\text{MCD}}(\tilde{\boldsymbol{x}}_j' \boldsymbol{v})} \tag{6.5}$$

   where $\hat{\mu}_{\text{MCD}}$ and $\hat{\sigma}_{\text{MCD}}$ are the univariate MCD estimators of location and scale. The set $\boldsymbol{B}$ consists of all directions $\boldsymbol{v}$ passing through two data points (or a random subset of these directions if n is very large).

   Thereafter, the $h_0 = \lceil \alpha n \rceil + 1$ observations with smallest outlyingness are considered, they are mean-centered and SVD is applied to them to find the k-dimensional subspace most closely to them (in L_2-norm). Here, the scree plot can be used to find an appropriate value for k, or the cumulative percent variation (CPV). For example, one could select k such that $\text{CPV} = \sum_{j=1}^{k} s_j^2 / \sum_{j=1}^{p} s_j^2 \geqslant 80\%$ with s_j the singular values of the SVD decomposition. Next, following Engelen et al. (2005), given the orthogonal distances to the preliminary subspace, we consider all observations with ODs smaller than the corresponding cut-off (as explained in Section 6.2.3). This yields an outlier-free index set H_1 of size h_1, which typically will be larger than h_0, in particular when α is chosen much smaller than the real proportion of regular observations.

2. Whereas ROBPCA applies CPCA on the observations from H_1, ROSPCA now uses sparse PCA. More precisely, we first standardize the data points of $\boldsymbol{X}$ with indices in H_1 using the componentwise median and the Q_n. Performing sparse PCA on them, by means of the Grid-based implementation of SCoTLASS with sparsity parameter λ, yields the sparse loadings matrix $\boldsymbol{P}_1 \in \mathbb{R}^{p \times k}$.

   We then perform an additional reweighting step that incorporates information about the sparse structure of the data, forming a bridge between the sparse and robust components of the algorithm and increasing efficiency. We discard variables with zero loadings on all k PCs and we

then compute the orthogonal distances to the estimated sparse PCA subspace. This yields an index set H_2 of observations with orthogonal distance smaller than the cut-off corresponding to these new orthogonal distances. We now standardize the subset of $\boldsymbol{X}$ with indices in H_2 using the componentwise median and the Q_n of the observations in H_1 (we use the same standardization as in the first time sparse PCA is applied). Then, sparse PCA is applied onto them, again by means of the Grid-based implementation of SCoTLASS with sparsity parameter λ. To get a full loadings matrix $\boldsymbol{P}_2$, we also need to add zero rows for all discarded variables to the estimated loadings matrix. The k-dimensional scores after reweighting are then given by $\boldsymbol{T} = (\boldsymbol{X} - \mathbf{1}_n\hat{\boldsymbol{\mu}}_1')\boldsymbol{P}_2$, with $\hat{\boldsymbol{\mu}}_1'$ the median of the observations in H_1. Intuitively, the goal of this reweighting is to recapture information from observations that are only outlying due to their behavior on variables that are found to be unimportant in our model, and use this information to obtain better estimates of the loadings corresponding to the important variables. Such observations will still have high OD values since the variables on which they are outlying will be compared to zero loadings in $\boldsymbol{P}_2$.

3. Finally, the eigenvalues are estimated robustly by applying the Q_n^2 estimator on the scores of the observations with indices in H_2. We need to use a robust measure of scale because observations with low OD and high SD that are included can influence the eigenvalue estimation. In order to robustly estimate the center, we compute the score distances and look at all observations of H_2 with a score distance smaller than the corresponding cutoff, this is the set H_3. We then estimate the center by the mean of these observations which gives the final center $\hat{\boldsymbol{\mu}}$ and the final scores $\boldsymbol{T} = (\boldsymbol{X} - \mathbf{1}_n\hat{\boldsymbol{\mu}}')\boldsymbol{P}_2$. We finally recompute the estimates of the eigenvalues by computing the sample variance of the (new) scores of the observations with indices in H_3 (the observations with low OD and high SD are not included anymore). The eigenvalues are sorted in descending order, so the order of the PCs may change. The columns of the loadings and scores matrices are changed accordingly.

Note that when it is not necessary to standardize the data, we only center the data as in the scheme above, but do not scale them.

6.2.6 Selection of sparsity parameters

SRPCA, SCoTLASS and ROSPCA use a scalar sparsity parameter λ in the Grid algorithm. Croux et al. (2013) select λ using a BIC (Bayesian Information Criterion) type criterion. It looks at the ratio of residual variances and the

degree of sparsity of the loadings matrix. These residual variances are computed by applying the Q_n^2 estimator to the sums of the squared OD statistics of the sparse and unconstrained PCA models. However, in our simulations and real data examples, this BIC approach selects λ values that are noticeably too sparse for ROSPCA, so we only use it for SRPCA. We choose λ by minimizing a BIC (Bayesian Information Criterion) type criterion based on the conventional formulation derived to use the Residual Sum of Squares (RSS). Our BIC-type criterion is:

$$\mathrm{BIC}(\lambda) = \ln\left(\frac{1}{h_1 p}\sum_{i=1}^{h_1} \mathrm{OD}_{(i)}^2(\lambda)\right) + \mathrm{df}(\lambda)\frac{\ln(h_1 p)}{h_1 p}, \tag{6.6}$$

where h_1 is the size of H_1, and $\mathrm{OD}_{(i)}(\lambda)$ is the ith smallest orthogonal distance for the model when using λ as the sparsity parameter. This criterion is similar to the BIC in regression, with the PCA orthogonal distances in place of the regression residuals. In ordinary regression, the residuals are univariate. Because the ODs are norms of p-dimensional vectors, we have to include p in (6.6). Moreover we use h_1 instead of n as this denotes the size of an outlier-free subset which does not depend on λ. After reweighting, if contamination is not high, h_1 is often close to n. Similar to Croux et al. (2013), $\mathrm{df}(\lambda)$ is taken as the number of non-zero loadings when λ is used as the sparsity parameter.

The first part of the criterion measures the quality of the fit whereas the second term penalizes for model complexity, reflecting a trade-off between accuracy and sparsity. In practice, we select λ by minimizing the BIC over the interval $[0, \lambda_{\max}]$ where $\lambda_{\max}$ gives full sparseness (exactly one non-zero loading per PC). We do this by looking at a grid of (usually equidistant) λ values over this interval.

Note that the computation of the index set H_1 in ROSPCA (step 1) does not depend on the choice of the sparsity parameter. It is therefore not necessary to run the full method each time we compute the BIC for a certain λ value. We perform the parts that are independent of λ only once and we then use this, for each value of λ we look at, as input for the parts that depend on the sparsity parameter (steps 2 and 3). This approach reduces the computation time and can lead to a considerable speed-up if many λ values need to be evaluated. This computational improvement cannot be applied to the SRPCA and SCoTLASS methods because in that case the Grid algorithm fully depends on the value of λ.

The computation time of ROSPCA is the result of its initial outlier detection part (step 1) and the remaining steps 2 and 3 to obtain sparsity. Figure 6.1 displays the computation times in seconds of ROSPCA (left) and SRPCA (right) for a range of values of n and p, and for $k = 2$ and $\lambda = 0$ using R 3.1.1 R

Core Team (2014) on Windows 7 (64-bit) OS with an Intel Core i7-3770 CPU @ 3.40GHz. The ROSPCA plot contains a further breakdown of computation time between the sparse and total computation times. The difference is the computation time attributable to the outlier detection step, which becomes more time consuming as n increases. Both ROSPCA and SRPCA show an increase in computation time as a function of n and p. The effect is noticeably stronger though for SRPCA, which shows much higher computation times as a function of both parameters (note the difference in the y-axis). This is primarily due to the way that the methods achieve robustness. ROSPCA performs a single outlier detection step, and then in the following steps it calculates the computationally inexpensive standard deviation for each direction in the Grid algorithm. In contrast, SRPCA relies on the comparatively slower Q_n statistic because robustness is achieved at the same time as sparsity is imposed. Note that the computation time of SRPCA is independent of the sparsity parameter λ. For ROSPCA, the computation time will decrease with λ since for higher values of λ, more variables can be excluded in the additional reweighting step which decreases the computation time of the second execution of SCoTLASS. We used $\lambda = 0$ to construct Figure 6.1, so computation times are lower when more sparsity is imposed using a higher value of λ.

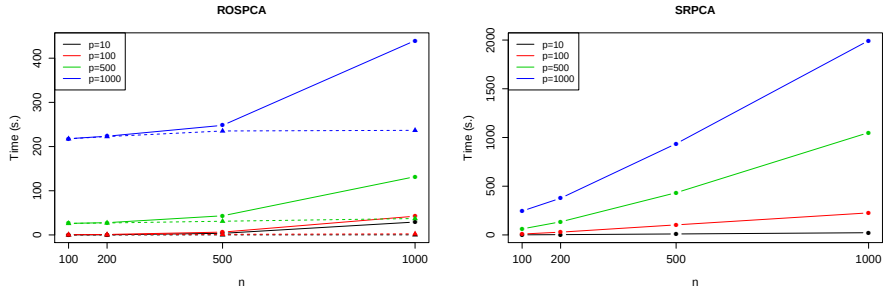

Figure 6.1: Computational performance of ROSPCA (left) and SRPCA (right) for varying values of n and p. The ROSPCA plot displays both the sparse (dashed line) and total (solid line) computation times.

6.3 Simulations

6.3.1 Layout of the simulation study

To evaluate the robustness, accuracy and sparsity of ROSPCA, we compare its performance with that of SRPCA, SCoTLASS, CPCA and ROBPCA on outlier-free and contaminated data. In specifying our simulations, we generate data from a multivariate normal distribution with a covariance matrix that has sparse eigenvectors. A varying proportion of the observations are replaced with outliers in order to test the robustness of the methods. We first standardize the data so that performing CPCA results in computing the eigenvectors (and -values) of the correlation matrix. Therefore, we need to generate a correlation matrix with sparse eigenvectors. First, we give a detailed description of the setup. Next, we evaluate the accuracy of the different PCA methods on the simulated data using performance measures based on the estimated loadings.

Let $\mathbb{R}^p$, with $p \geqslant 8$, be our original data space, and let $k = 2$ be the number of important components. We generate a correlation matrix such that it has sparse eigenvectors. We design the correlation matrix to have 3 groups of variables with no correlation between variables from different groups. The first two groups consist of b variables each, where b is an integer that we choose to be at least 4. The correlation between the different variables of the group is equal to $a_1 \in [-1, 1]$ for group 1 and $a_2 \in [-1, 1]$ for group 2. The third group contains the remaining $p - 2b$ variables, which we specify to be uncorrelated. Our correlation matrix $\boldsymbol{R}$ is thus equal to

$$\boldsymbol{R} = \begin{pmatrix} \boldsymbol{R}(a_1) & \mathbf{0}_{b\times b} & \mathbf{0}_{b\times(p-2b)} \\ \mathbf{0}_{b\times b} & \boldsymbol{R}(a_2) & \mathbf{0}_{b\times(p-2b)} \\ \mathbf{0}_{(p-2b)\times b} & \mathbf{0}_{(p-2b)\times b} & \boldsymbol{I}_{p-2b} \end{pmatrix}$$

with $\boldsymbol{R}(x)$ the $b \times b$-matrix with ones on the diagonal and off-diagonal elements $x \in [-1, 1]$, and $\boldsymbol{I}_{p-2b}$ the $(p - 2b)$-dimensional identity matrix. When $a_1 > a_2$, the first two sparse eigenvectors are given by $\boldsymbol{p}_1 = -\frac{1}{\sqrt{b}}\boldsymbol{q}_1$ and $\boldsymbol{p}_2 = -\frac{1}{\sqrt{b}}\boldsymbol{q}_2$ with $\boldsymbol{q}_1 \in \mathbb{R}^p$ a vector with the first b elements equal to one and zero elsewhere, and $\boldsymbol{q}_2 \in \mathbb{R}^p$ a vector with the second b elements equal to one and zero elsewhere. The first b variables should therefore have zero loadings for the second PC, and similarly for the next b variables and the first PC. It is also clear that the variables from the last group should have zero loadings for both PCs. The order of the first two eigenvectors is changed when a_1 is smaller than a_2. The statements about the zero loadings can be adapted accordingly. Note that the eigenvectors are, neglecting their order, independent of the choice of a_1 and a_2.

Next, the correlation matrix $\boldsymbol{R}$ is transformed into the covariance matrix $\boldsymbol{\Sigma} = \boldsymbol{V}^{\frac{1}{2}}\boldsymbol{R}\boldsymbol{V}^{\frac{1}{2}}$, where $\boldsymbol{V}$ is the diagonal matrix containing the variances of the variables to be detailed later. The n observations are generated from a p-variate normal distribution with mean $\mathbf{0}$ and covariance matrix $\boldsymbol{\Sigma}$. Standard normally distributed noise terms are also added to each of the p variables to make the sparse structure of the data harder to detect. This gives a dataset $\boldsymbol{X} = \boldsymbol{X}_u + \boldsymbol{X}_{noise}$ with $\boldsymbol{X}_u \sim N_p(\mathbf{0}, \boldsymbol{\Sigma})$ and $\boldsymbol{X}_{noise} \sim N_p(\mathbf{0}, \boldsymbol{I}_p)$. Finally, $100\varepsilon\%$ of the data points are randomly replaced by outliers. We consider different proportions of outliers, namely $\varepsilon = 0, 0.1, 0.2, 0.3, 0.4$. These outliers are generated from a p-variate normal distribution $N_p(\boldsymbol{\mu}_{out}, \sigma^2_{out}\boldsymbol{I}_p)$ with $\boldsymbol{\mu}_{out} = 25(0, -4, 4, 2, 0, 4, -4, 2, 3, -3, \ldots, 3, -3)'$ and $\sigma^2_{out} = 20$, as in Croux et al. (2013). Importantly, these outliers do not follow the correlation structure determined by $\boldsymbol{R}$. They will therefore bias non-robust sparse methods trying to estimate the sparse structure. We also denote the dataset with the outliers by $\boldsymbol{X}$.

First, we consider a low-dimensional setting with $p = 10$ dimensions and $b = 4$ in our simulations, so we have two blocks of four useful variables and the last two variables are noise. We take $a_1 = 0.9$ and $a_2 = 0.5 < a_1$ which gives eigenvalues $3.7, 2.5, 1, 1, 0.5, 0.5, 0.5, 0.1, 0.1, 0.1$ and the first two eigenvectors of $\boldsymbol{R}$ are given by $\boldsymbol{p}_1 = -\frac{1}{2}(1, 1, 1, 1, 0, 0, 0, 0, 0, 0)'$ and $\boldsymbol{p}_2 = -\frac{1}{2}(0, 0, 0, 0, 1, 1, 1, 1, 0, 0)'$. Importantly, the difference between the first and second eigenvalue is large enough such that the methods can clearly determine that $\boldsymbol{p}_1$ is the loading vector of the first PC. When taking a_1 and a_2 closer together, the difference between the first two eigenvalues gets smaller, so it becomes more difficult for the PCA method to identify which of the first two eigenvectors corresponds to the first PC. We also need to make sure that a_2 is large enough, otherwise the difference between the second and third eigenvalue is too small. This can again cause problems because the PCA method can sometimes select the third eigenvector as the loading vector corresponding to the second PC, making our bias criterion become difficult to interpret. With our choices for a_1 and a_2, the difference between the eigenvalues is large enough to avoid these problems. We take $\boldsymbol{V} = \text{diag}(100, \ldots, 100, 25, \ldots, 25, 4, 4)$, so the variables in a group have the same variance. For each simulated scenario, we generate 500 datasets following the above scheme to thoroughly characterize the behavior of the methods.

Figure 6.2 shows a heat map of the absolute values of one dataset from our simulation setting with $p = 10$, $n = 100$ and $\varepsilon = 0.2$. The outliers are visible as the observations with values taking a dark blue color. Despite being fairly easy to identify on a heat map, we shall see that these can pose difficulties for sparse PCA methods that are not highly robust. We note that the configurations we use to evaluate the methods considered in the paper are known to be particularly challenging for them, while they are capable of easily identifying outliers in

other configurations that are not clearly revealed by a heat map.

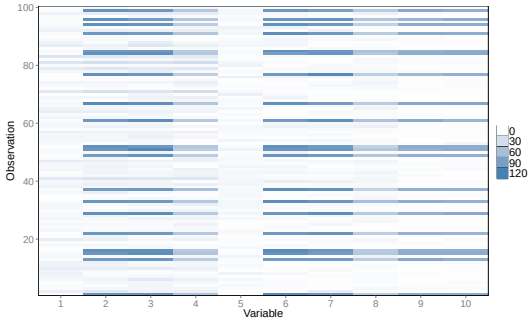

Figure 6.2: Heat map of absolute value of simulated data with $p = 10$, $n = 100$ and $\varepsilon = 0.2$. Outliers are visible in dark blue.

We also look at a high-dimensional setting with $p = 500$ and $k = 2$. In contrast to the low-dimensional setting, the first two groups consist of $b = 20$ variables each, which results in 40 useful variables and 460 noise variables. In the new setting, the eigenvalues are $18.1, 10.5$, 1 (460 times), 0.5 (19 times) and 0.1 (19 times), where we take $a_1 = 0.9$ and $a_2 = 0.5 < a_1$ again. The first two sparse eigenvectors are given by $\boldsymbol{p}_1 = -\frac{1}{\sqrt{20}}\boldsymbol{q}_1$ and $\boldsymbol{p}_2 = -\frac{1}{\sqrt{20}}\boldsymbol{q}_2$ with $\boldsymbol{q}_1 \in \mathbb{R}^{500}$ a vector with the first 20 elements equal to one and zero elsewhere, and $\boldsymbol{q}_2 \in \mathbb{R}^{500}$ a vector with the second 20 elements equal to one and zero elsewhere. We use the same variances for the groups as before: 100 for group 1, 25 for group 2 and 4 for group 3. For each scenario, we now generate 100 datasets following the high-dimensional scheme to keep computations reasonable.

To compare the robustness of the methods, we look at the 2nd principal angle between the subspace spanned by the two dominant eigenvectors of the correlation matrix $\boldsymbol{R}$ and the subspace spanned by the columns of the estimated loadings matrix (the PCA subspace), as was also done in Hubert et al. (2005) and Todorov and Filzmoser (2013). We compute this angle using the algorithm of Björck and Golub (1973). This angle lies between 0 and $\frac{\pi}{2}$, and we divide it by $\frac{\pi}{2}$ to get values between 0 and 1. In the remainder we will refer to the standardized version as the "angle". It is clear that we want values close to 0.

All simulations were performed in R 3.1.1 using following functions: `prcomp` (CPCA), `PcaHubert` (ROBPCA) from the *rrcov* package [Todorov and Filzmoser (2009)] and `SPcaGrid` (SRPCA and SCoTLASS) from *rrcovHD* [Todorov (2014)]. We used self-written functions for ROSPCA based on the code for `PcaHubert`. For ROSPCA and ROBPCA the parameter α is set to 0.5, yielding maximal robustness. First, we compare the estimation of the PCA subspace and the

degree of sparsity attained. Then, we discuss the behavior of the λ selection step of these algorithms following our BIC criterion (6.6) for ROSPCA and SCoTLASS, and the BIC criterion of Croux et al. (2013) for SRPCA.

6.3.2 Results of the simulation study

6.3.2.1 Subspace estimation

We start with the low-dimensional simulations ($p = 10$). For each simulation setting and each sparse method we report two results as boxplots. On the left is a boxplot of the angle values corresponding to a model fitted by a method with λ selected using the previously discussed criteria. We consider following grid of λ values: $\{0, 0.02, \ldots, 2.5\}$. The boxplot on the right is based on the minimal angle value attained by each method over the same range of λ values. These results provide two insights. First, the boxplot based on the minimal angle values gives a sense of the performance of each method if λ were selected to give the fit closest to the real structure of the data possible for that method. Secondly, this boxplot and the boxplot to its left, based on results from models using λ values selected by a criterion, together give a sense of how successful the information criterion is in selecting an optimal value of λ for the method. For CPCA and ROBPCA, we only have the boxplot of the angle values corresponding to the fitted model.

Figures 6.3, 6.4 and 6.5 show boxplots on datasets of increasing size n and contamination rate ε. Mean values are indicated with blue diamonds. As expected, bias decreases and the angles become less dispersed when n increases. SCoTLASS reports the best results for $\varepsilon = 0$ but performs very badly when contamination is present. Also of note, the boxplots corresponding to models based on selected λ values are only slightly higher than the boxplots based on the minimal angle values, showing that the λ selection problem is tractable for SCoTLASS under these settings. Over all contamination levels, ROSPCA shows a low mean and median bias, even for the case where $\varepsilon = 0.4$. Like SCoTLASS when it is applied to uncontaminated data, the boxplots based on selected λ values and the minimal angles tend to be close, meaning that for ROSPCA, λ is typically selected accurately. At small sample sizes, quite some variability is still present in the estimates, but this decreases substantially at larger sample sizes. In contrast to ROSPCA, SRPCA returns distinctly higher biases, even for the best possible λ value. Its bias at outlier-free data only becomes reasonably small when n is very large. Furthermore, the difference in the boxplot pairs for SRPCA reveals that the BIC selection criterion proposed by Croux et al. (2013) yields angles that are on average quite distinct from the optimal ones that could

be obtained. CPCA is outperformed by the sparse methods SCoTLASS and ROSPCA at outlier-free data, and completely breaks down at contaminated ones. ROBPCA shows an increased bias when contamination is present. A closer look at the results revealed that the method did correctly identify the outliers, but it was not able to discover the sparse structure of the data as well as ROSPCA does.

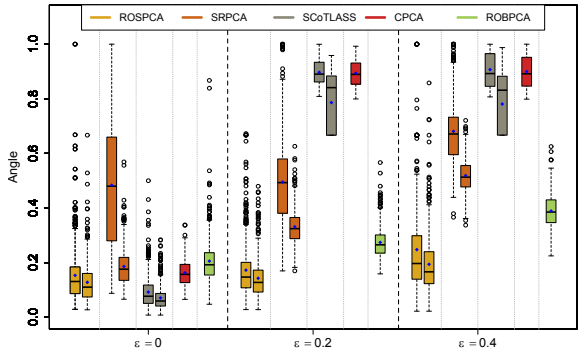

Figure 6.3: Angle values of ROSPCA, SRPCA, SCoTLASS, CPCA and ROBPCA at $p = 10$ and $\varepsilon = \{0, 0.2, 0.4\}$ for $n = 50$.

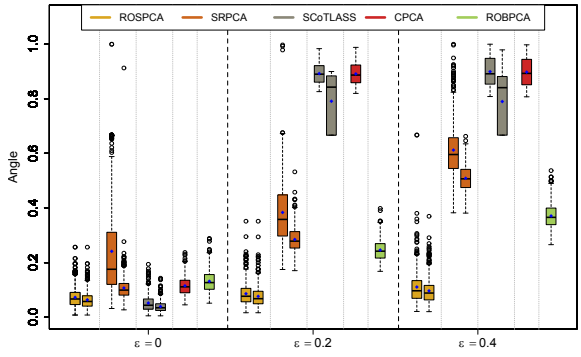

Figure 6.4: Angle values of ROSPCA, SRPCA, SCoTLASS, CPCA and ROBPCA at $p = 10$ and $\varepsilon = \{0, 0.2, 0.4\}$ for $n = 100$.

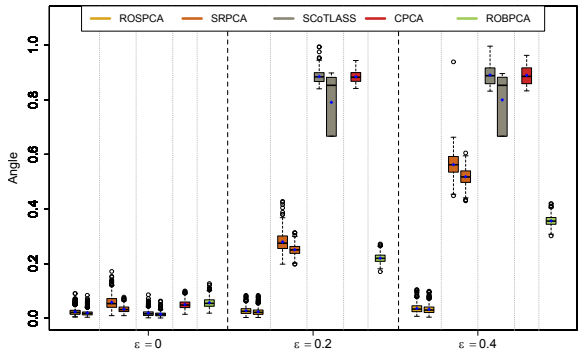

Figure 6.5: Angle values of ROSPCA, SRPCA, SCoTLASS, CPCA and ROBPCA at $p = 10$ and $\varepsilon = \{0, 0.2, 0.4\}$ for $n = 500$.

Consider now the high-dimensional simulations where $p = 500$. We now consider the following grid of λ values: $\{0, 0.02, \ldots, 1.2\}$. For SRPCA with $n = 500$, we decreased the grid with λ values up to 0.6 instead of 1.2 to keep computations reasonable. Figures 6.6, 6.7 and 6.8 show the results for several sample sizes. As before, the bias and the dispersion of the angle becomes smaller when the sample size n increases. On uncontaminated data, the selection of λ is not successful for SCoTLASS (the BIC from Croux et al. (2013) returns even slightly worse results). However, the minimum angle boxplot shows that SCoTLASS can perform well, and ROSPCA attains similar performance to SCoTLASS's optimal performance in both boxplots. SRPCA shows very poor performance even when outliers are not present when λ is selected, and has worse results for the minimal angle values as well, indicating that intrinsically it may not be as accurate as SCoTLASS or ROSPCA. CPCA and ROBPCA have a comparable behavior, which is inferior to the sparse methods. When contamination is introduced, SCoTLASS performs very poorly, as expected, while the optimal performance of SRPCA and ROSPCA is only slightly worse than when the data is not contaminated, and ROSPCA continues to show successful λ selection. When $\varepsilon = 0.4$, SRPCA does however show higher bias than for lower ε, unlike ROSPCA. CPCA is no longer reliable, whereas the performance of ROBPCA remains stable.

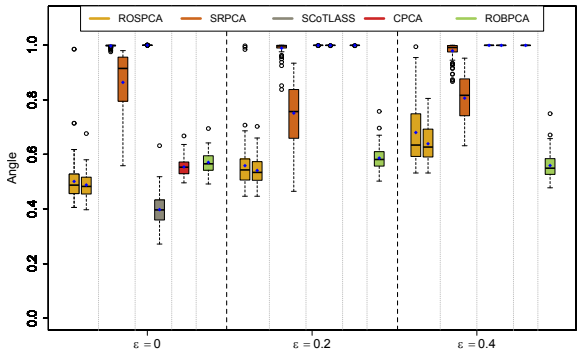

Figure 6.6: Angle values of ROSPCA, SRPCA, SCoTLASS, CPCA and ROBPCA at $p = 500$ and $\varepsilon = \{0, 0.2, 0.4\}$ for $n = 50$.

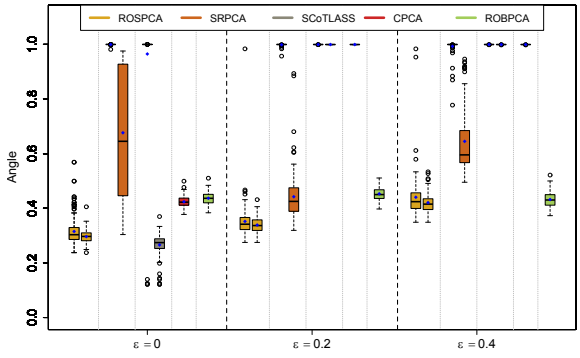

Figure 6.7: Angle values of ROSPCA, SRPCA, SCoTLASS, CPCA and ROBPCA at $p = 500$ and $\varepsilon = \{0, 0.2, 0.4\}$ for $n = 100$.

6.3.2.2 Sparsity

In addition to estimating a model that is not influenced by outliers, it is also important to estimate the correct sparsity. The *zero measure* is one way to compare how correctly each of the methods estimates the sparse $\boldsymbol{P}$. For each element of $\boldsymbol{P}$, it is equal to one if the estimated and true value are both zero or both non-zero, and 0 otherwise. We then take the average zero measure over

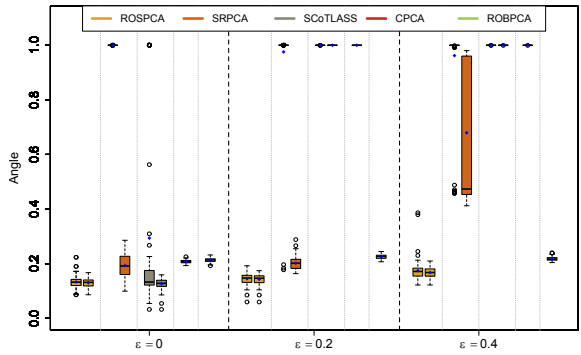

Figure 6.8: Angle values of ROSPCA, SRPCA, SCoTLASS, CPCA and ROBPCA at $p = 500$ and $\varepsilon = \{0, 0.2, 0.4\}$ for $n = 500$.

all elements of $\boldsymbol{P}$ and all 500 simulations which we call the *total zero measure*. We need to specify when an element is "equal to zero" because it can be that an element of $\boldsymbol{P}$ is very small but different from zero. We say that all elements with an absolute value smaller than 10^{-5} are "equal to zero".

In Figures 6.9, we see that ROSPCA accurately discerns the sparse structure of $\boldsymbol{P}$, even when $n = 50$ and $\varepsilon = 0.4$. SRPCA steadily demonstrates weaker performance as ε increases, whereas SCoTLASS performs well for $\varepsilon = 0$, and uniformly poorly for higher values of ε. The zero measure plots for larger sample sizes are very similar to the plot for $n = 100$. These results show that ROSPCA not only gives robust PCA estimates but is also better at detecting the sparse structure of the data. CPCA and ROBPCA hardly yield zero loading elements, so their zero measure is almost constantly equal to 40%, which is the percentage of non-zero entries in $\boldsymbol{P}$.

The zero measure is less useful in the high-dimensional setting because perfect sparsity for all zero loadings is more difficult to achieve. This results in zero measures that are comparatively more difficult to interpret than those shown in Figure 6.9, since two methods may appear to give similar results by this measure, while a close inspection of the loadings reveals substantial differences.

6.3.2.3 The λ selection performance of ROSPCA, SRPCA and SCoTLASS

As explained in Section 6.2.6, we use the BIC-type criterion (6.6) to select the sparsity parameter λ of ROSPCA and SCoTLASS (since no criterion is

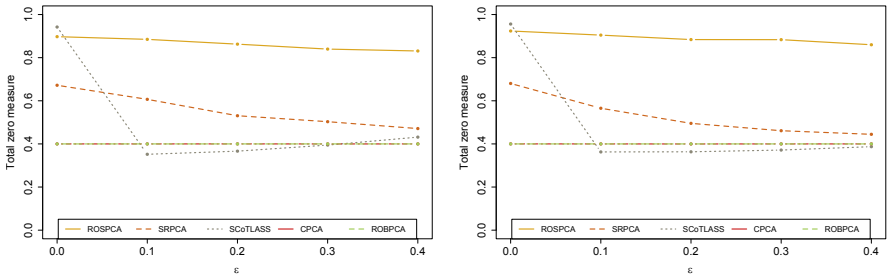

Figure 6.9: Total zero measure of ROSPCA, SRPCA, SCoTLASS, CPCA and ROBPCA for (left) $n = 50$ and (right) $n = 100$.

proposed in Jolliffe et al. (2003)). For SRPCA, we use the BIC proposed by Croux et al. (2013). We looked at 101 (equidistant) values of λ over the interval in which complete sparsity is attained: $[0, 2.5]$, i.e. $\{0, 0.02, \ldots, 2.48, 2.5\}$. To provide insight into the role of robustness in this process, we introduce $\varepsilon = 20\%$ contamination. In Figure 6.10, we display the quantile plots of the angle values obtained by these methods over the 500 simulated datasets for $n = 100$ and $\varepsilon = 0.2$ as a function of λ. It depicts the median (solid lines) and first (dotted lines) and third quartile (dashed lines) of angle values for a given λ value over the 500 simulations.

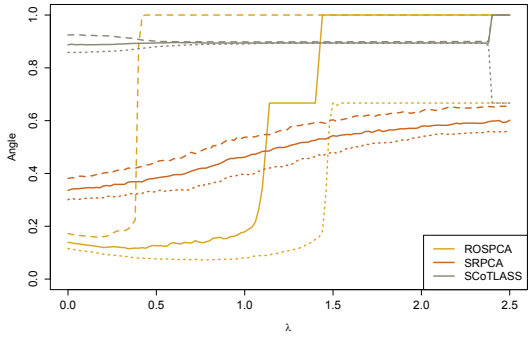

Figure 6.10: Quantile plots of the angle values for ROSPCA, SRPCA and SCoTLASS as a function of λ.

Examining the angle values corresponding to fits for each of the methods using different values of λ reveals a pattern correlated with the robustness of the

methods. The angle values for SCoTLASS, tend to be fairly constant and high across the range of λ. This reflects the fact that the models are all influenced by outliers, and in comparison the sparsity of the model has very little impact of the angle. The quantile plot for SRPCA is not as flat as that of SCoTLASS and is considerably lower, but shows a steadily increasing angle value as λ is increased. Since this method is robust, it can attain decent fits with non-sparse models, but including sparsity makes it vulnerable to missing the outliers and finding a worse fit. This has the consequence that even though the true data is sparse, a full SRPCA model attains the lowest angle value since it allows for the most accurate outlier screening.

The quantile plot for SRPCA illustrates a trade-off between robustness and sparseness, where we find that contamination due to outliers tends to dominate the inaccuracy due to using a non-sparse model on sparse data (which is why the full SRPCA model has the lowest angle). The ROSPCA quantile plot shows that it is possible to account for both the sparse structure of the data and the outliers. For ROSPCA, the lowest value of λ (0 in our case) does not correspond to the lowest angle value. Rather, this is achieved by a sparse model, as we would expect. This is possible because ROSPCA has initially separated the outlier detection and sparsity steps before combining insights from both to return the final model. The first and third quantiles show that there is some variation in the angle values returned by ROSPCA for different values of λ, but the figures in Section 6.3.2.1 show that the value of λ selected by the BIC criterion is consistently close to the value of λ returning the minimal angle for each simulation.

6.4 Real data example

In this section we illustrate the behavior of ROSPCA and SRPCA on the glass dataset introduced in Hubert et al. (2005). It consists of Electron Probe X-ray Microanalysis (EPXMA) spectra over $p = 750$ wavelengths and 180 collected glass samples [Lemberge et al. (2000)]. Although the non-sparse ROBPCA performs well on this dataset, employing a sparse method may be interesting because when one consults the full loadings, the data actually appears to have a sparse structure. Figure 6.11 shows a heatmap of the absolute values of the centered data matrix where we used the componentwise median. We only plotted the wavelengths with numbers 120-400 because the rest of them are mostly non-informative (due to the sparse structure of the data). As noted in Hubert et al. (2005), two groups of outliers can be clearly identified in this dataset: the last 38 observations that were measured after the spectrometer

was cleaned and calcium outliers with high values for two groups of wavelengths between 300 and 370.

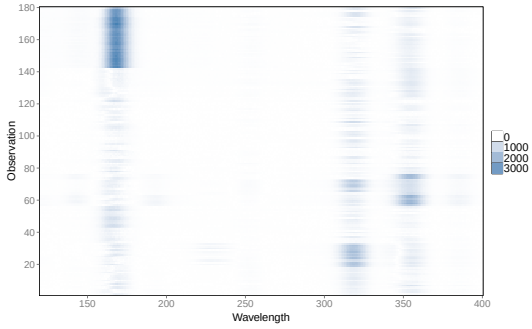

Figure 6.11: Heat map of the Glass data.

With a sparse robust PCA analysis we hope to achieve outlier detection results comparable to ROBPCA while also obtaining sparse loadings that reflect the atomic structure of the glass samples. We do not standardize the data because all variables are expressed in the same units. The non-robustness of SCoTLASS means that it cannot reliably address the outliers present, so results are omitted.

Selecting the number of components to use in a sparse PCA model is more complicated than in classical PCA due to the inclusion of λ, which varies with k, but must also be selected. In Jolliffe et al. (2003), rather than providing a criterion for selecting k that accounts for sparsity, the authors apply the cumulative percent variation (CPV) criterion to a non-sparse PCA model. Then, they discuss the influence of a range of λ values over a model using that particular value of k. In Croux et al. (2013), the authors fit a robust, non-sparse PCA model with many components and then use those eigenvalues to select k for the sparse, robust model. Similarly, we use the eigenvalues of the robust, non-sparse PCA model described in Step 1 of ROSPCA. Since the SVD is computed on uncontaminated observations, we obtain eigenvalues for all possible $\min(p, n-1)$ components. We use the scree plot corresponding to these eigenvalues to select the number of components to retain, but automatic criteria such as the CPV can also be used.

The scree plot for ROSPCA (Figure 6.12) indicates that three or four components are sufficient to model the data well, and we select four components. Additionally we set the parameter $\alpha = 0.5$ to obtain maximal robustness. Hence $h_0 = \lceil 0.5 \times 180 \rceil + 1 = 91$. We also select $k = 4$ for SRPCA after consulting the scree plot for SRPCA with $\lambda = 0$.

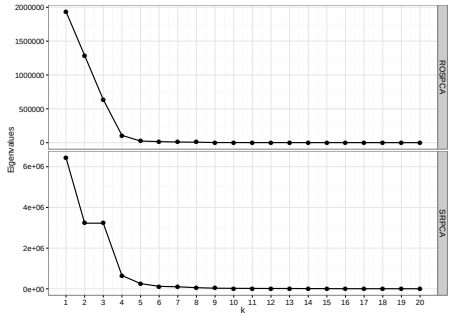

Figure 6.12: Scree plots for ROSPCA and SRPCA ($\lambda = 0$).

Next, we perform the λ selection step for ROSPCA using our proposed BIC, and for SRPCA using the BIC of Croux et al. (2013). This yields λ values of 0.96 and 72.7, respectively. The running time for ROSPCA using $\lambda = 0.96$ was 146s, whereas SRPCA had a running time of 419s. For comparison we also include the ROBPCA results. As its scree plot is identical to that of ROSPCA (since the singular values are computed on the same subset of observations), we also use $k = 4$ components.

From the fitted models we can produce outlier maps showing the score distance and orthogonal distance of the observations in the dataset. We normalize these diagnostic plots by dividing each of the distances by its cut-off to make the results visually comparable across methods. This gives us Figure 6.13. All three methods indicate the post-cleaning observations (orange) as bad leverage points, but SRPCA does not show the same discriminatory power as ROSPCA and ROBPCA. These two methods also clearly find several other orthogonal outliers and bad leverage points. This is useful for the practitioner because it provides a clear message that these observations warrant further investigation. Ignoring the boundary cases, we have indicated this set of outliers, as detected by ROSPCA, as open blue circles. Obviously ROBPCA identifies these outliers as well, but SRPCA rather declares them as ambiguous border cases with only larger score distances. Next, we compared the heatmap of the data in Figure 6.11 with these outlier maps, and noticed that almost all open blue circles correspond to calcium outliers which were highlighted on the heatmap. The three open blue circles that are close to the cut-off line for the score distances on the diagnostic plot of ROSPCA are however not clearly visible on the heatmap. Only a closer inspection of the raw data revealed that they are outlying on variables 215–245. Our robust multivariate analysis was able to detect this abnormal behavior at once.

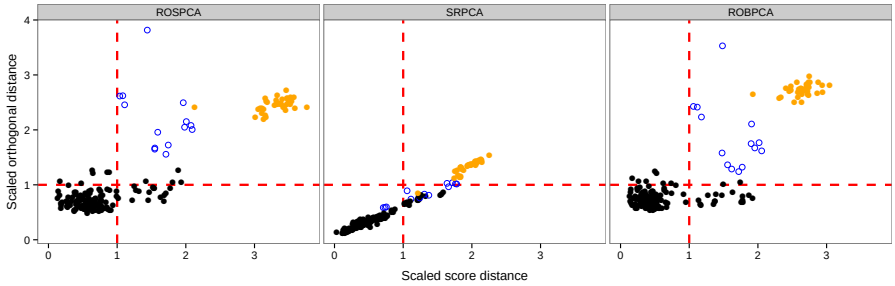

Figure 6.13: Scaled outlier maps of ROSPCA (with $\lambda = 0.96$), SRPCA ($\lambda = 72.7$) and ROBPCA on the glass data. The orange points correspond to the measurements after the window has been cleaned. The open blue circles correspond to the other outliers identified by ROSPCA.

To study the sparsity, we plot the loadings of each of the methods in Figure 6.14 and tabulate the sparsity of each in Table 6.1. Unsurprisingly, ROBPCA produces the least sparse loadings, with only 13 variables with all loadings less than the threshold of 10^{-5}. Nonetheless, the loadings are instructive as they give a sense of the full structure of the data and where sparsity might be obtained. Specifically, three groups of wavelengths (155–185, 310–335, 336–370) are particularly relevant. SRPCA attains the greatest sparsity, but given the poor outlier detection performance, it is likely that as we saw in the simulation studies, the λ selection procedure has been influenced by contamination. The sensitivity of the λ selection step to outliers underscores the need for a highly robust method. ROSPCA obtains loadings similar to those of ROBPCA, but with the important distinction that loadings ROBPCA assigned small values to are now assigned no weight, resulting in 200 excluded variables. This increases the interpretability of the resulting model, while retaining accuracy. We note that a practitioner may choose a larger λ in an ad hoc way to further increase the sparsity of ROSPCA and that for a value of λ giving similar sparsity to that of SRPCA, ROSPCA still identifies the outliers correctly.

Finally, we also compare the obtained loadings using the angle measure, results are shown in Table 6.2. We see that the ROSPCA and ROBPCA subspaces are similar and that the SRPCA subspace differs a lot from the other two subspaces. One could also visually deduce these conclusions from inspecting Figure 6.14.

The results for the glass dataset reinforce our findings from the simulations. Since the outliers are in two groups, we find that SRPCA does well at detecting the more obvious post-cleaning ones, but struggles to find the more nuanced

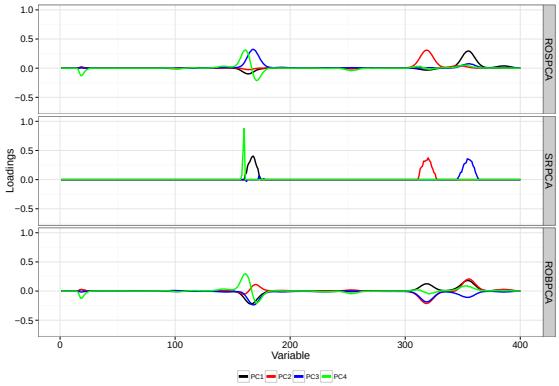

Figure 6.14: Loadings of ROSPCA (with $\lambda = 0.96$), SRPCA ($\lambda = 72.7$) and ROBPCA on the glass data. Loadings on wavelengths with indices above 400 were small for all methods and are excluded from the plot.

Table 6.1: Number of non-zero loadings (larger than 10^{-5}) for each method per PC. The bottom row is the number of variables that have zero loadings (smaller than 10^{-5}) on all 4 PCs.

	ROSPCA	SRPCA	ROBPCA
PC1	359	14	733
PC2	272	17	735
PC3	491	34	737
PC4	408	4	736
No. of excluded variables	200	696	13

Table 6.2: Angle between the obtained loadings for the Glass data using ROSPCA, ROBPCA and SRPCA.

	ROSPCA-ROBPCA	ROBPCA-SRPCA	SRPCA-ROSPCA
Angle	0.040	0.731	0.725

calcium outliers. As in the simulations, ROSPCA both detects the outliers accurately and finds a plausible sparse structure.

6.5 Discussion

ROSPCA and extensions have the potential to tie together two issues that arise in modern process monitoring. First, as discussed in Chapter 2, outliers must be removed during the initial fitting of the monitoring model. In Chapter 5, this guideline was applied in pre-processing the stamping data. The second issue is that many modern monitoring applications contain redundant sensor data. Variable selection can be useful when monitoring these types of process data streams because it improves interpretation and can reduce the risk of over-fitting. ROSPCA is a step towards a general approach for preparing fault free calibration data for process monitoring models. However, to fully address this issue, it is desirable that extensions can account for the non-stationarity that can sometimes arise in the calibration data. Doing so would make it possible to use larger calibration windows when parameterizing the monitoring methods, which should result in more effective monitoring performance in some scenarios.

6.6 Conclusions

We have detailed a new approach for sparse, robust Principal Component Analysis, ROSPCA, that is a modification of ROBPCA. Unlike existing methods for sparse PCA, ROSPCA prioritizes the detection of the outliers rather than giving robustness and sparsity equal weight. Our results indicate that this approach is warranted. We observe that by first detecting and neutralizing the outliers, ROSPCA is able to fit the sparse structure of the majority of the data with high accuracy. In comparisons with existing methods, we find that ROSPCA consistently obtains the best performance.

In addition to good robustness and sparsity properties, ROSPCA is also computationally faster. One of the most important steps in performing a robust PCA analysis is the selection of the λ parameter. A single execution of ROSPCA is faster than one of SRPCA, but this advantage is compounded when selecting λ since the robustness step only needs to be performed once.

Chapter 7

Conclusion and future perspectives

This dissertation explored SPM approaches for high-dimensional, time-dependent processes, and in particular PCA-based methods. The context in which this dissertation finds itself is one in which classical latent variable methods have been used successfully to monitor high-dimensional processes without time dynamics, but methods capable of modeling processes with this characteristic remain less well understood. The existing literature concerned with this setting has not sufficiently detailed the relative merits of PCA-based SPM methods, nor has it provided sufficient guidance for their implementation.

7.1 Chapter 2

In Chapter 2, a survey of SPM methods for high-dimensional processes with a focus on those capable of monitoring time-dependent processes is conducted. While a range of methods are mentioned, most attention is paid to PCA-based methods and their application. In addition to discussing important variants of PCA for autocorrelated and non-stationary processes, an overview of their extensions and ancillary analytical techniques are covered. This chapter highlighted possibilities for the practitioner faced with challenging monitoring problems, but stayed close to the claims made by the literature. A number of questions were raised in this chapter, pointing to directions for future research.

Control charts based on Static PCA models have been widely used for monitoring

systems with many variables that do not exhibit autocorrelation or non-stationary properties. DPCA, RPCA, and MWPCA provide methodologies for addressing these scenarios. To summarize, a rubric of the situations where these methods are applicable is provided in Table 7.1. However, while extensions have sought to make them as generally implementable as Static PCA, a number of challenges have not yet been resolved.

Table 7.1: Applicability of different PCA methods to time-dependent processes.

		Non-Stationarity	
		No	Yes
Auto correlation	No	Static PCA	R/MWPCA
	Yes	DPCA	?

An area for further research lies in investigating the performance of models mixing DPCA and R/MWPCA to handle autocorrelation and non-stationarity simultaneously. Presently, works have focused on examining the performance of methods intended for only one type of dynamic data, but combinations of the two remain unexplored.

Currently, a weakness of DPCA is that if an observation is considered out-of-control, but as an outlier rather than a fault, then the practitioner would normally continue monitoring, but ignoring this observation. However, doing so destroys the lag structure of DPCA. Therefore, a study on the benefits of reweighting the observation like in Choi et al. (2006), or removing the observation and replacing it with a prediction would be a useful contribution.

Further research is also warranted in the area of fault isolation. The contribution plot, residual-based tests, and variable reconstruction are three well-studied approaches for solving this problem [Kruger and Xie (2012); Qin (2003)]. Recently, some new methods for fault isolation based on modifications to the contribution plot methodology have been proposed (see Elshenawy and Awad (2012)). However, these methods cannot isolate the source of faults in many complex failure settings; a task which becomes more difficult still when the data is time-dependent. Improvements on the classical contribution plot or entirely new methods would be a valuable addition to the PCA control chart toolbox. Woodall and Montgomery (2014) cover some control chart performance metrics, such at the average run length and FDR, and elaborate on challenges faced by these metrics in real-data applications. They propose that the FDR may be more appropriate for high-dimensional cases, but state that further research is necessary to draw firm conclusions. This advice is especially relevent for PCA

control chart methods, since they are often applied to high-dimensional data, and the FDR should be investigated as an option for measuring performance.

7.2 Chapters 3 and 4

In Chapter 3, claims made in the literature on PCA-based monitoring were examined more closely on very challenging monitoring scenarios. In an extensive comparison study focusing on fault detection capability, core methods in the PCA-based toolbox were applied on high-dimensional, time-dependent processes with different properties. In Chapter 4, a spectrum of difficulties was considered for the different process types to identify when the where monitoring methods are successful and unsuccessful at modeling processes. The results of these chapters provide a more nuanced account of these methods than that of the literature. Many claims about the behavior of the methods were found to hold, but others are not supported by our results. For example, classical PCA and DPCA, a method designed to cope with autocorrelated processes that challenge for classical PCA were found to perform similarly. For most process types the study revealed that PCA-based methods tend to be more accurate at identifying one of either score or sensor faults, and that a combination of methods may be necessary to detect both. Finally, these studies highlighted the challenges faced in parametrizing these methods. The adaptive PCA methods proved to be particularly difficult to fit models for, and a general conclusion was that more concrete guidelines needed to be formulated to support practitioners interested in applying these methods.

Further research should focus on a rigorous approach for setting control limits when the monitored series is time-dependent. With correct definitions of the control limits, more complex comparative analyses, such as the Average Run Length, would be possible. We note that for some of the processes studied, even the simplest metrics, such as the detection rates, are highly variable despite considering many simulated realizations. This phenomenon is mostly a result of the different behavior of the time-dependent data at different realizations, since the simulated deviations do not always have the same impact on the observed data. For all of the scenarios, we found it necessary to adjust the control limits, and α values to achieve the desired false detection rate. This introduces additional complexity for model specification and furthermore, we saw that good modeling performance on NOC data may not be enough to guarantee good fault detection properties.

Furthermore, the control limits of RPCA and MWPCA are problematic. We currently use fitted values of α in the classic control limit formulas for the

monitoring statistics, but these α values do not directly correspond to intuitive values. Further research might investigate the use of control limits that are based on alternative distributional approximations, or a robust updating approach that prevents faulty observations from contaminating the estimated covariance matrix.

7.3 Chapter 5

Chapter 5 was concerned with addressing the last two topics from the previous section. It provides a detailed examination of the consequences of poor model fitting, and proposes methods both for choosing the forgetting factors defining the updating behavior of the adaptive methods, and secondary tuning of the control limits since even a model with adaptive properties rarely provides a perfect model of complex processes. Simulations and real data examples are used to illustrate how these approaches can be applied in practice. Some directions for future work on this topic are highlighted below.

It would be interesting to know more about the equivalence of RPCA and MWPCA models. It is possible to find a relationship between an exponentially weighed moving average and a windowed average, so one would suspect that a similar relationship can be obtained for these two PCA methods.

We believe that extensions to adaptive PCA-based methods, such as the use of a subspace modeling step to increase interpretability, could still make use of the parameter selection procedures we propose. Another possible use comes from Choi et al. (2006) and He and Yang (2008), who proposed adaptive forgetting parameters to account for changes in the non-stationarity properties of the process. Based on these approaches, the forgetting factor changes to reflect changes in the estimated process mean and covariance structure. This may lead to a more accurate model, and improved monitoring statistics, but it still requires a well chosen initial value for the forgetting parameter. The approach we outlined may be used to select this initial value. The guidelines given in this chapter could also form the basis for parameter selection for adaptive PLS methods.

The monitoring procedures discussed in this chapter all start from calibrating and validating a NOC data set. To eliminate possible outliers in this initial data set, one can first apply a robust PCA method as we illustrated in Section 5.4.2. Doing so, parameter estimates are obtained which are not unduly influenced by outliers. These robust estimates are then used as starting values for the updating procedure, which ignores new faults through an accurate selection of the control limits which we developed in this chapter. Alternatively one could

apply robust PCA methods throughout the whole test monitoring phase. This is however much more time-consuming since fast online updates of robust methods are not (yet) available. Computational techniques, such as those developed in Engelen and Hubert (2005) and Hubert and Engelen (2007), first need to be developed to obtain a procedure that can be applied in practice. This will be part of future research.

7.4 Chapter 6

In earlier chapters, the topic of outliers in the calibration set arises. Typically, one assumes that this data set is fault free. However, this assumption is often violated. In these cases, a method from robust statistics can be used to remove outliers in a systematic fashion. Chapter 6 introduced a new approach to robust, sparse PCA based on the ROBPCA method. An advantage of this approach is that it can simultaneously remove outliers from the calibration data set while focusing the practitioner on important variables and latent structures in the data. The capacity to reveal latent structures more clearly than classical robust PCA is particularly relevant in process monitoring applications because these can correspond directly to well defined components of the process that are otherwise difficult to discern in the mass of data modern sensor systems generate.

The work in this chapter opens the door to the development of sparse robust methods for high-dimensional data, such as sparse robust discriminant analysis, sparse partial least squares regression, and for skew-adjusted sparse PCA. Extensions of the ROBPCA based methods, as in Vanden Branden and Hubert (2005), Hubert and Vanden Branden (2003) and Hubert et al. (2009) will be studied. A theoretical study of the influence function of ROSPCA, extending the results of Debruyne and Hubert (2009), is also an interesting challenge for future research.

Appendix

A HDCC - The Matlab Toolbox for Multivariate and High-dimensional Control Charts

Joint work by Rato, T. and Schmitt, E.

In this appendix, we introduce a new Matlab toolbox, **HDCC** (High-dimensional Control Charts), that implements multivariate and high-dimensional control charts. This toolbox is intended for both practitioners and researchers, providing a starting environment for familiarization with such monitoring procedures, as well as a code basis to begin investigation of standing problems in this class of control charts. The code of this toolbox was developed and test with release R2010a of Matlab.

The rest of this appendix is organized as follows. In Section A.1, a brief review of the theory of the control chart methods included in this toolbox is provided. In Section A.2 and A.3, the functions contained in our toolbox **HDCC** are described. In Section A.4, we apply these methods to the classic example of the Tennessee Eastman data set.

A.1 Control charts for processes with many variables

Processes range in dimensional complexity from one variable to thousands of variables. The appropriate monitoring strategies change over this spectrum as the rising number of variables cause violations to the assumptions of classes of monitoring methods, such as multi-collinearity. Many users of this toolbox will be familiar with univariate methods, such as the exponentially weighted moving average, and their multivariate extensions. We will first introduce these methods, which we have implemented with the toolbox, before turning to our

main focus, which is latent variable-based monitoring methods that are capable of handling extremely high-dimensional processes.

A.1.1 Multivariate control charts

In addition to PCA-based monitoring, HDCC also supports some multivariate control charts. Since these have not been covered earlier in the dissertation, some background is required. Two commonly used multivariate techniques to monitor small deviations on the mean value are the multivariate cumulative sum (MCUSUM) and multivariate exponentially weighted moving average (MEWMA) control charts. These control charts aim to introduce more information about the monitoring process and thus to improve their detection capabilities. In order to do so, the MCUSUM control charts resort to the application of successive sum of the measurements vector that are above a reference or allowance value, v, related with the smallest deviation that we are interested in detecting. For the case of the MCUSUM proposed by Crosier (1988), this summation is performed by defining,

$$\boldsymbol{C}_i = \begin{cases} \mathbf{0} & \text{if } d_i \leqslant v \\ (\boldsymbol{C}_{i-1} + \boldsymbol{x}_i - \boldsymbol{\mu}_0)\left(1 - \frac{v}{d_i}\right) & \text{otherwise} \end{cases}$$

$$\text{with } d_i = \sqrt{(\boldsymbol{C}_{i-1} + \boldsymbol{x}_i - \boldsymbol{\mu}_0)' \, \boldsymbol{\Sigma}_0^{-1} \, (\boldsymbol{C}_{i-1} + \boldsymbol{x}_i - \boldsymbol{\mu}_0)} \tag{1}$$

where $v > 0$ is a reference value, $\boldsymbol{C}_0 = \mathbf{0}$, $\boldsymbol{\mu}_0$ is the $(p \times 1)$ target vector and $\boldsymbol{\Sigma}_0$ is the invertible $(p \times p)$ in-control covariance matrix. The parameters $\boldsymbol{\mu}_0$ and $\boldsymbol{\Sigma}_0$ are typically set using data exhibiting normal operating conditions. The monitoring statistic is then computed as $MCUSUM_i = \sqrt{\boldsymbol{C}_i^T \boldsymbol{\Sigma}_0^{-1} \boldsymbol{C}_i}$ and is compared against a suitable control limit. Likewise, the MEWMA control chart proposed by Lowry et al. (1992) introduces past information by application of a weighted window that gradually gives lower weights to past observations according to,

$$\boldsymbol{z}_i = \lambda \boldsymbol{x}_i + (1 - \lambda) \, \boldsymbol{z}_{i-1}$$

where $0 \leqslant \lambda \leqslant 1$ is a forgetting parameter and $\boldsymbol{z}_0 = \mathbf{0}$. Then, a Hotelling's T^2 like statistic can be computed as,

$$T_i^2 = \boldsymbol{z}_i' \boldsymbol{\Sigma}_{\boldsymbol{z}_i}^{-1} \boldsymbol{z}_i' \tag{2}$$

where the covariance matrix is given by,

$$\boldsymbol{\Sigma}_{\boldsymbol{z}_i} = \frac{\lambda}{2 - \lambda}(1 - (1 - \lambda)^{2i})\boldsymbol{\Sigma}_0$$

The functions `mcusumModel` and `mcusumMonitor` fit an MCUSUM model and monitor a process with it. Similarly, the functions `mewmaModel` and `mewmaMonitor` apply for the MEWMA control chart.

A.1.2 Static PCA

The functions `pcaModel` and `pcaMonitor` fit a PCA model and monitor a process with it. Control limits for the monitoring statistics can be set using theoretical derivations or based on empirical adjustments on a training data set. Details on the basic concepts of Static PCA and its extensions are discussed in Chapter 2

A.1.3 Selection of the number of components

To select the number of components to retain in PCA-based models, one can resort to several methods (see e.g. Valle et al. (1999) and Jolliffe (2002)). Component selection is performed in the function `pcaModel()`, with the Cumulative Percentage of Variance (CPV), cross-validation, and parallel analysis made available.

The CPV is a measure of how much variation is captured by the first k latent variables:

$$\text{CPV}(k) = \frac{\sum_{j=1}^{k} \lambda_j}{\sum_{j=1}^{p} \lambda_j} 100\%.$$

The number of latent variables is selected such that the CPV is greater than the minimum amount of variation the model should explain. To use cumulative percentage of variance, set the option `kSelect` to `kSelectCPV` in `pcaModel()`.

Parallel analysis, proposed by Horn (1965), performs an eigenvalue decomposition on the original data matrix and on an uncorrelated data set with the same dimensions. Then the eigenvalues of the two decompositions are compared, and the number of retained latent variables corresponds to the last point where the eigenvalues of the original data are larger than the eigenvalues for the uncorrelated data. The eigenvalues of the uncorrelated data set will all tend to one as n grows, but for smaller sample sizes the first components will be larger than one, and the last components will be smaller. The purpose of generating a data set with the same dimensions as the original data is to account for this. Since parallel analysis requires knowledge of the size of the data set on which the model is trained, and this is not precisely known for RPCA (see Chapter 2 Section 2.5)due to the use of exponential forgetting of observations, it is not implemented for this method. To use parallel analysis, set the option `kSelect` to `kSelectParAnl` in `pcaModel()`.

A cross-validation approach for selecting the number of components based on the work of Krzanowski and Kline (1995) is implemented. This method randomly divides the data set $\boldsymbol{X}$ of n observations and m variables into g groups. The most intuitive implementation is that for a given k, g models are estimated, one on each data set obtained by excluding one of the groups. These models are then used to predict the values in the group that was missing from the data set they were estimated on, and the differences between the predicted and actual values yield n errors. Based on these errors, the Predicted Error Sum of Squares (PRESS) can be computed. However, one can take advantage of the fact that $PRESS(k) \approx (\lambda_{k+1} + \ldots + \lambda_m)/nm$, to more efficiently compute the PRESS. In this scheme, k is varied from 1 to m, and the approximate PRESS values are computed for all values of k. Krzanowski and Kline (1995) argue that the value of k maximizing the following function is a good choice for the number of latent variables:

$$\left(\frac{PRESS(k-1) - PRESS(k)}{PRESS(k-1) - PRESS(k+1)}\right)\left(\frac{n+m+2k-1}{2n+2m-4k-2}\right). \qquad (3)$$

To use cross-validation, set the option `kSelect` to `kSelectCrossVal` in `pcaModel()`.

A.1.4 Dynamic PCA (with Decorrelated Residuals)

Dynamic PCA (DPCA) was first proposed in Ku et al. (1995) as a way to extend static PCA to autocorrelated, multivariate systems.

The functions `dpcaModel` and `dpcaMonitor` fit a DPCA model and monitor a process with it.

Introducing lagged variables in DPCA allows the description of the autocorrelation present in the data. However, the T^2 and Q statistics can still exhibit autocorrelation. In the case where enough lags are selected, the Q-statistic should indeed have no serial correlation. Yet, even in this case, there is autocorrelation in the scores, and subsequently in the T^2-statistic. To overcome this issue, Rato and Reis (2013c) proposed a combination of DPCA with a missing data estimation technique [Nelson et al. (1996); Arteaga and Ferrer (2002)] in order to obtain better time-decorrelated statistics that they call DPCA-DR.

The functions `dpcadrModel` and `dpcadrMonitor` fit a DPCA-DR model and monitor a process with it. Within these functions, `missingData` applies the missing data estimation method used by Rato and Reis (2013c) to the first m variables in $\mathbf{X}$ based on a DPCA model.

A.1.5 Selection of the number of lags for dynamic PCA methods

In order to specify the number of lags, Rato and Reis (2013b) detail an approach for selecting the number of lags by variable, allowing for a more refined model of the process being monitored. This lag selection algorithm makes use of the ability of singular vectors to describe linear relationships when the singular values are small. Similarly, dynamic relationships can also be captured by adding lagged variables. As these small singular values relate to the variance of prediction errors, the equivalent linear relationships are stronger for smaller singular values. Based on these concepts, the lag selection algorithm performs a step wise search in order to determine the combination of lagged variables that leads to an augmented data set, with minimum singular values. To do so, in each stage of the procedure, the m variables are tested, one at a time, for the addition of one more lag than in the lagged structure of the previous stage. For each combination, the respective singular values are computed and the lagged variable that leads to the smallest singular value is kept in the augmented matrix. Furthermore, the smallest singular value is saved as the Key Singular Value (KSV) of the stage. Note than even though all variables are tested, only one is kept in the final lagged structure of the stage. This procedure is repeated until a pre-defined upper limit on the number of lags is achieved. Optimal lagged structures are located in stages where the KSV decays more or less sharply and then becomes approximately constant. To better capture this behavior the Ratio of successive Key Singular Values ($KSVR$) is computed. Following these two criteria, the selected lagged structure should have a low KSV (meaning that the augmented data can describe dynamic relationships) and low $KSVR$ (indicating that a significant decrease in the KSV has just occurred). The KSV and $KSVR$ are combined in an optimization function (ϕ) used to find the lagged structure that match both conditions. With this additional criterium available, one can select the number of lags simply by finding out the minimum ϕ. The `lagSelect()` function performs the above procedure and generates graphical representations of the decision parameters for lag selection.

A.1.6 Adaptive PCA methods

The functions `rpcaModel` and `rpcaMonitor` fit an RPCA model and monitor a process with it. Similarly, the functions `mwpcaModel` and `mwpcaMonitor` fit a MWPCA model and monitor a process with it. The majority of the concepts from Static PCA carry over to these methods. However, an important distinction is that RPCA and MWPCA adjust for process non-stationarity by including new observations and downweighting old ones. The rate at which observations are forgotten is determined by the selected η value for RPCA, and H value

for MWPCA. Chapter 5 treats this topic in detail and provides the following approach to parameter selection. The objective of including new observations and forgetting old ones is to model the non-stationary process accurately. This accuracy of the model can be quantified through the Q-statistic, which gives a sense of how large the difference between the model estimates and observed values are. A good value of the forgetting parameter will correspond with low Q-statistic values, which we quantify using the sum of Q-statistics from a validation data set. Selection of the forgetting parameter is simplest when the sum as a function of the parameter has a clear minimum. Another possibility is that the sum of Q-statistics will converge to a low value as the forgetting parameter increases, and that we will not observe a minimum. Such a result would indicate the need for a large forgetting parameter, but this may have an adverse impact on the detection power of the T^2. In this case, one may wish to select a value of the forgetting parameter which is large enough to obtain a low model error, but not so large that the T^2-statistic is compromised. One heuristic approach for doing this is to select the forgetting parameter corresponding to an elbow of the error curve. Selection of η and H is performed using `fSpeedSolve()`. This function provides recommendations of the forgetting parameter obtaining the minimum error and also lets the user manually select a value.

A.1.7 Control limits

Analytical or empirical control limits can be developed for the monitoring statistics of the PCA methods described above. Assuming temporal independence and multivariate normality of the scores, the $100(1-\alpha)\%$ control limit for Hotelling's T^2 is

$$T^2_\alpha = \frac{k(n^2-1)}{n(n-k)} F_{k,n-k}(\alpha). \tag{4}$$

Here, $F_{k,n-k}(\alpha)$ is the $(1-\alpha)$ percentile of the F-distribution with k and $n-k$ degrees of freedom. If the number of observations is large, the control limits can be approximated using the $(1-\alpha)$ percentile of the χ^2 distribution with k degrees of freedom, thus $T^2_\alpha \approx \chi^2_k(\alpha)$. The simplicity of calculating this limit is advantageous.

HDCC uses the approximation for the control limit of the Q-statistic derived from the general result of Box Box (1954), which shows that the sum of squares, and therefore the Q-statistic, is approximately distributed as $g\chi^2(h)$. Provided that all the eigenvalues of the matrix $\boldsymbol{S}$ are available, the parameters are given

by:

$$\theta_i = \sum_{j=k+1}^{p} \lambda_j^i \text{ for } i = 1, 2;\ g = \frac{\theta_2}{\theta_1};\ \text{and } h = \frac{\theta_1^2}{\theta_2}$$

The functions `limitT` and `limitQ` set theoretical control limits for the T^2 and Q-statistics for PCA-based models. These take a value of α and k as inputs. The eigenvalues are also used as input by `limitQ`. These functions can be used to establish limits for all PCA methods, though they are currently used only for RPCA and MWPCA, as empirical limits are preferred for the non-adaptive PCA-based models. For adaptive methods, updating the control limits is necessary as the number of latent variables vary, and the underlying mean and covariance parameters of the PCA model also change. In order to do so, for the T^2, it is only necessary to recalculate $T^2_\alpha = \chi^2_k(\alpha)$ for the newly determined number of latent variables, k_t. Furthermore, since $Q(\alpha)$ is a function of θ_i which are in turn functions of the eigenvalues of the covariance matrix, once the new PCA model has been estimated, the Q-statistic control limit is updated to reflect changes in these estimates.

In the presence of autocorrelation and non-stationarity, the distributional assumptions made on the monitoring statistics by the analytical expressions for control limits are often violated. For this reason, empirical control limits estimated based on the distributions of the monitoring statistics on training and validation data set are used. It is possible to estimate fully empirical control limits for PCA, DPCA and DPCA-DR since these methods operate under the assumption that the process is stationary and therefore they use a fixed model and control limit throughout all the monitoring period. Empirical control limits for these methods can be estimated using `modelLimits()`. This function takes a `pcaModel`, or analogous output from another model function, a matrix $\boldsymbol{X}$ of NOC data and the desired global False Detection Rate (FDR) as input, and returns control limits giving the desired FDR.

A different approach is necessary to ensure that the adaptive methods also achieve the intended false detection rate on NOC data since the control limits are not constant. To do so, we search for values of α for the T^2 and Q (call these α_{T^2} and α_Q), which, when used as input in the analytical expressions for the control limits of these statistics, result in an FDR_{T^2} and an FDR_Q whose sum equals the desired total FDR for the model. To accomplish this, we impose that FDR_{T^2}=FDR_Q, and assume that T^2 and Q are independent, which may be violated in practice but nonetheless typically leads to well chosen values. Note that the adjusted values of α_{T^2} and α_Q do not correspond to the statistical significance of the monitoring statistics, since the theoretical expressions for the

control limits are not valid if the modelling assumptions are violated. Further, they need not to be equal to each other. To select the tuned value of the α parameters for the T^2 and Q-statistics the function `aSolve()` is used. The resulting values are then inputed to `limitT` and `limitQ` in order to compute the control limits.

Once the T^2 and Q-statistics and their limits have been calculated, they can be plotted onto a control chart. **HDCC** comes with a number of functions that call model functions and produce charts and statistics as output. These are:

- `controlChart`: Calls modeling and monitoring functions and allows the user to apply these methods and obtain control statistics and control charts as output.
- `GUIcontrolChart`: GUI interface for using `controlChart`.
- `plotChart`: Function called by `controlChart` to create control chart plots.

Examples illustrating their use will be provided in the following section.

A.2 Simulating high-dimensional, time-dependent process data

HDCC comes with the capability to simulate high-dimensional versions of a number of common process-types from time series analysis. The intention is to provide researchers with a testing ground for their own methods (which can, of course, be compared against benchmark methods already implemented in **HDCC**), and to give practitioners a transparent context to experiment with methods that may be suitable for their applications.

To obtain each observation at time t we began by generating k scores, $\boldsymbol{y}_t$, according to the equation of the desired process. For all process types, we introduce variation onto the process dynamics through $\boldsymbol{\varepsilon} \sim \mathcal{N}(\boldsymbol{0}_k, 0.01\boldsymbol{I}_k)$, where $\boldsymbol{I}_k$ is a $k \times k$ identity matrix. These are then transformed into a p-dimensional data set of measurements computed as

$$\boldsymbol{x}_t = \boldsymbol{y}_t \boldsymbol{P}_0' + \boldsymbol{e}_t, \tag{5}$$

where $\boldsymbol{P}_0$ is a $p \times k$ orthogonal, randomly generated matrix. The $\boldsymbol{e}_t$ are $p \times 1$ vectors of white noise errors, distributed as $\mathcal{N}(\boldsymbol{0}_p, 0.005^2\boldsymbol{I}_p)$, that simulate measurement noise, as is done, for instance, in (Ku et al. (1995) and Lakshminarayanan et al. (1997)). The $\boldsymbol{e}_t$ can be seen as the error at the

sensor level, and are set to a small value here under the assumption that sensors are typically reliable. To reproduce results, it is necessary to save the generated this generated data.

HDCC allows users to generate six different types of processes and a variety of fault types. The AR(1) process is defined as (Box et al. (1994)):

$$\boldsymbol{y}_t = \phi \boldsymbol{y}_{t-1} + \boldsymbol{\varepsilon}_t, \tag{6}$$

where $\boldsymbol{y}_t$ are the serial observations of the underling latent model ($\boldsymbol{Y}$ in Equation (5)) and ϕ is the AR coefficient. Box et al. (1994) defines the MA(1) process as given in Equation (3.4) where φ is the MA coefficient. Box et al. (1994) defines the ARI(1,1) as given in Equation (3.5). (Box et al. (1994)) defines the IMA(1,1) process as given in Equation (3.6). Although not explicitly given as an option, the user can also generate an I(1) process by setting the φ parameter to zero to obtain:

$$\boldsymbol{y}_t = \boldsymbol{y}_{t-1} + \boldsymbol{\varepsilon}_t. \tag{7}$$

Finally, **HDCC** can also generate data that is non-stationary on the subspace of the process (NSS). The NSS process is non-stationary locally, but exhibits a periodic fluctuation that can be considered stationary on the larger scale. The NSS process expressing the described behavior introduces non-stationarity in the form of rotations on the base latent variables hyperplane, $\boldsymbol{P}_0$. By application of such rotations, the latent variables hyperplane experiences a periodic fluctuation over all its axes. In this case we set the amplitude to $15\pi/180$, which corresponds to a $\pm 15°$ rotation on the base hyperplane, and the frequency to $1/1000$ (i.e., a full rotation is obtained at every 1000 observations).

For each of these six process types, the user can introduce a fault at the score or the measurement level. **HDCC** can generate both step and ramp faults. Step faults apply the entire magnitude of the fault to all observations following the beginning of the fault. The ramp fault increases the magnitude of the fault linearly from zero to the selected magnitude from the introduction of the fault until the end of the data set on which the fault was applied.

Faults are introduced to the process on either the first score in $\boldsymbol{y}_t$, or the first measurement variable (sensors) in $\boldsymbol{x}_t$, by simple addition of a step deviation with magnitude defined as d times a reference standard deviation. This reference standard deviation can be either based on the error components ($\boldsymbol{\varepsilon}_t$ and $\boldsymbol{e}_t$) or the signal ($\boldsymbol{y}_t$ and $\boldsymbol{x}_t$). The latter will give results in line with deviations to the total variation of the process, which is the common set-up in textbook examples. Setting the magnitude of fault deviations based on the error components is useful for comparing detection performance across different process types since

this allows that if two different processes are well modelled, then what remains to create variation in the modelling statistics are the terms ($\boldsymbol{\varepsilon}_t$ and $\boldsymbol{e}_t$), which are set to be comparable.

A.3 The Matlab toolbox

In this section we provide an overview of the functionality of **HDCC**. **HDCC** comes with data generation tools so that high-dimensional, time-dependent data for a number of process types can be created to user specification. This is intended to provide a test bed for practitioners to build insight and well-defined data streams for researchers to evaluate methods on. We first show how data can be generated using the data generation GUI (`guiTDseries`). Next we show how this data can be monitored with functions from the toolbox. We will focus on implementation of the control charts it creates through the GUI interface `GUIcontrolChart`, while accompanying this with the background code called by the GUI.

A.3.1 Data generation with guiTDseries

The `guiTDseries` function allows the user to create series of arbitrary dimensionality of the process types AR(1), MA(1), ARI(1,1), IMA(1,1), and NSS (the I(1) process can be generated via the ARI(0,1) or IMA(1,0)). The first four process types are canonical time series processes Box et al. (1994). More details on the NSS process are provided in Chapter 3, but in essence it produces rotations on the subspace defining the process behavior.

Upon calling `guiTDseries`, the following GUI appears (Figure 1).

A.3.1.1 The layout

Figure 1 shows the layout of the GUI after it has launched. The sections of the plot and corresponding options are:

- **Process type**: Select the type of process to generate. This can be an AR(1), MA(1), ARI(1,1), IMA(1,1), NSS, or the user can provide a saved process model from which parameters will be used to simulate the data.
  - `mu`: When simulating an ARI(1,1) or IMA(1,1) process, `mu` is added to the scores of the process. The default value is 0.

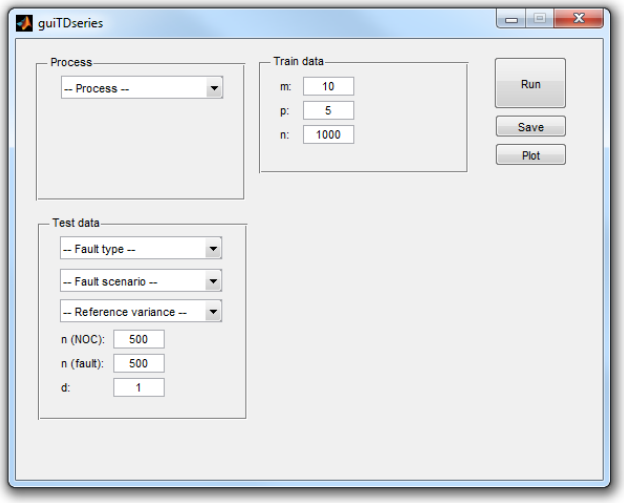

Figure 1: Screenshot of the `guiTDseries` GUI as it initially appears.

  - `phi`: Value for the ϕ parameter in the AR(1) and ARI(1,1) processes determines how much weight is given to the previous observation in the autoregressive component. In the MA(1) and IMA(1,1) processes ϕ determines how much weight is given to $\boldsymbol{\varepsilon}$ from the previous period.
  - `freq`: Rotation frequency of the NSS hyperplane. This value is related with the number of observations needed to perform a full rotation over all the axis of the base hyperplane. For instance, a value of 1/1000 means that a full rotation is obtained at every 1000 observations. Smaller values lead to slower rotations.

- **Train Data**: Set the number of observations `n`, number of variables `m`, dimensionality of the subspace `p` for the model training data set, and rotation frequency `freq` of the NSS model subspace.

- **Test Data**: Allows the user to define some characteristics of the faults.

  - `Fault type` Faults can be either sensor or score faults, meaning that they occur either on the variable level, or on the level of the subspace. If real faults are likely to occur due to sensor failure, or the failure of a specific component, then it is appropriate to simulate sensor faults. If real faults are likely to occur on the latent structures in the process, then score faults should be simulated.
  - `Fault scenario`: Determines whether the fault will be a step fault, meaning a sudden increase of size `d`, or a ramp fault in which the

process moves to a new state `d` standard deviations from the NOC level.

- `Reference variance`: Determines the reference variance to scale the fault's magnitude. This scaling can be done based on either the errors ($\boldsymbol{\varepsilon}_t$ and $\boldsymbol{e}_t$) or signals ($\boldsymbol{y}_t$ and $\boldsymbol{x}_t$) standard deviation.
- `n(NOC)` and `n(fault)`: Indicates how many observations in the test data set will be generated according to the NOC and fault conditions, respectively.
- `d`: Determines the magnitude of the fault as described in 3.4.

- **Run**: Runs the data generation function `TDseries`, creating a data test and training data sets according to the user's specifications.
- **Save**: Saves the training and test scores (`Ttrain` and `Ttest`) and the training and test values for $\boldsymbol{X}$ (`Xtrain` and `Xtest`). The model parameters used to generate the simulated data are also outputted as `model`.
- **Plot**: Plots the generated values of the scores and data for the training and test series.

A.3.1.2 Example: Generating an AR(1) series

To illustrate features of `GUIcontrolChart`, we will generate an AR(1) series with the following specifications:

- `mu` = 0.
- `phi` = 0.9.
- `n` = 1000.
- `m` = 100.
- `p` = 5.
- `Fault type` = "score".
- `Fault scenario` = "step".
- `Reference variance` = "Errors variance".
- `n (NOC)` = 500.
- `n (fault)` = 500.

- d = 10.

After generating data, with these properties, we can plot them and get the output shown in Figure 2.

A similar result can be obtained using the following Matlab commands:

```
% Set process parameters
nNOC=500; % Number of NOC observations
nFault=500 % Number of faulty observations;
m=100; % Number of measured variables
p=5; % Number of latent variables
T_params.phi=0.9; % Autoregressive coefficient
T_params.mu=0; % Mean values of the time series
A_params=[]; % Parameters for the NSS process
(not needed in this case)
scenarioProcess='AR'; % Type of process
fault.scenario='step'; % Fault's scenario
fault.type='score'; % Fault's type
fault.shift=10; % Fault's magnitude
fault.refSigma='error'; % Reference standard deviation
for the fault's magnitude
% Simulate process
output = TDseries(nNOC+nFault, m, p, T_params, A_params,
scenarioProcess, fault);
```

A.3.2 SPM with GUIcontrolChart

To exemplify the use of this toolbox for SPM, we will create static PCA control charts for the data we generated in the previous section. Calling the `GUIcontrolChart` function, we get the GUI shown in Figure 3

A.3.2.1 The layout

Figure 3 shows the layout of the GUI after it has launched. The sections of the plot and corresponding options are:

- **Data**: The user selects the calibration/validation and test data sets from the workspace for the model fitting, as described in Chapter 5, Section 5.2.
- **Monitoring Method**: The user selects the monitoring method they would like to use from the available methods. Parameter selection options

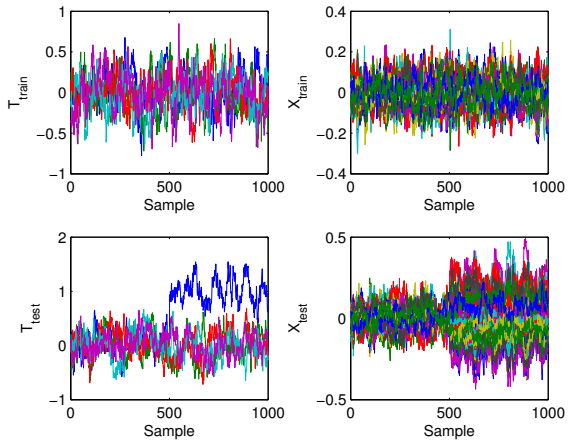

Figure 2: Plots of AR(1) scores and data with a step fault generated using `guiTDseries`.

will appear below for the selected method. The details for each method are explained in Section A.1. The user must select the desired FDR for their monitoring application and the approach used to set the control limits (either theoretical or empirical).

- **Options**: The user can choose to plot the monitoring online (real-time), or to provide a plot of the monitoring results after monitoring is complete. This final plot can also be set to use a log scale.
- **Run**: Runs the desired monitoring method on the calibration and test data sets provided by the user. If parameter values are not specified, and require user input, the user will be prompted to provide this input during running.
- **Save**: Saves the training and test scores (`Ttrain` and `Ttest`) and the training and test values for $\boldsymbol{X}$ (`Xtrain` and `Xtest`). The model parameters used to generate the simulated data are also outputted as `model`.
- **Plot**: Plots the generated values of the scores and data for the training and test series.

Depending on the monitoring method selected, different parameters need to be chosen. The **HDCC** toolbox focuses on methods for high-dimensional methods and offers a number of tools for automatically parametrizing the PCA models.

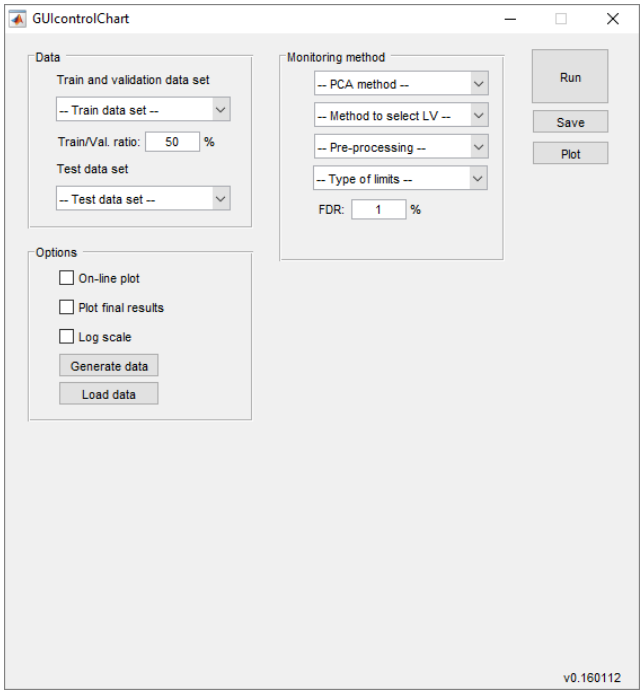

Figure 3: Screenshot of the `GUIcontrolChart` GUI as it initially appears.

Many of these are automatically performed once the selection methodology is specified. However, for the selection of the forgetting factor and lags required by some methods, additional steps are required by the user.

For methods requiring a forgetting factor, leaving the value for this parameter blank in the input box will result in automatic evaluation of candidate values. The user is then prompted to select a value for the forgetting parameter based on a curve of summed squared prediction errors on a validation set. In addition to the curve, **HDCC** also identifies the forgetting parameter corresponding to the minimum error over the range of the parameter that was searched. To illustrate this, we apply the RPCA method to our simulated data. In Figure 4, we show the error curve produced by **HDCC** and the interface allowing the user to select the forgetting parameter obtaining minimal error on the validation set, or to input a value of their choice.

DPCA and DPCA-DR models require the number of lags to be specified for each variable. **HDCC** can identify the number of lags automatically, following the methodology of Rato and Reis (2013b). Alternatively, the user can manually

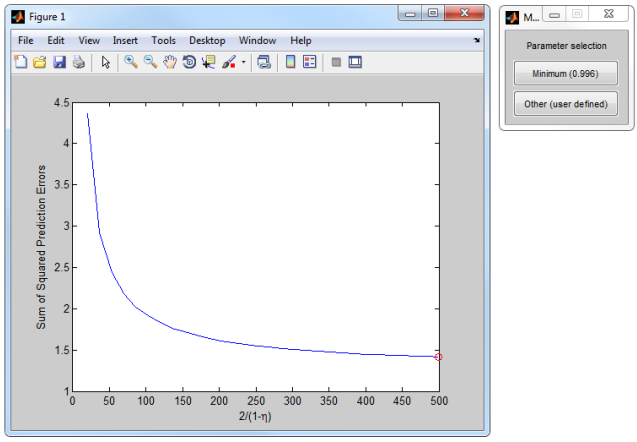

Figure 4: Screenshot of the squared prediction error curve used to select the forgetting parameter of RPCA.

select the number of lags. **HDCC** assists in this process by providing a plot of the singular value metrics (KSV and $KSVR$), which are described in detail in the real data example later in this paper. Regardless of whether the user chooses to automatically or manually select the number of lags, a maximum number of lags to evaluate for each variable must be specified. As the number of variables increases, the computing time needed to evaluate more complex lag structures grows as well.

A.3.2.2 Example: SPM using PCA

To perform SPM based on a static PCA model the user starts by loading the relevant data in the **Data** section (see Figure 3). In the `Training and validation data set` pop-menu, the workspace variables are listed and the user is prompt to select the data for modelling PCA, which in this case is `Xtrain`. This data set is partitioned into train (used for training the model) and validation (used for selecting the forgetting parameters and control limits) data according to `Train/Val. ratio`, where a 25% ratio implies 25% of the data is used in the training data set. Likewise, test data set, `Xtest`, can be loaded from the workspace.

In the **Monitoring method** section we define the modelling approach, "PCA", the method to select the number of principal components, `kSelectCpv`, and the scaling approach, `Auto-scaling`. As part of the model parametrization, the

FDR and CPV threshold are also set to default values. The control limits are set via an empirical approach.

Finally, in the **Options** section we choose to plot the monitoring results after completion. The final look of the GUI is presented in Figure 5. By running the monitoring procedure with these configurations, the control charts depicted in Figure 6 are obtained. The corresponding Matlab commands of this operation are:

```
% Set modelling parameters
aG = 0.01;% False detection rate
trainValRatio = 50; % Train to validation ratio
% Modelling
model = pcaModel( Xtrain(1:midPoint,:), 'kSelectCpv', 'datamat' );
% Set control limits
model = modelLimits( Xtrain(midPoint+1:end,:), model, aG );
% Monitoring
[ stat, ucl ] = pcaMonitor( Xtest, model );
```

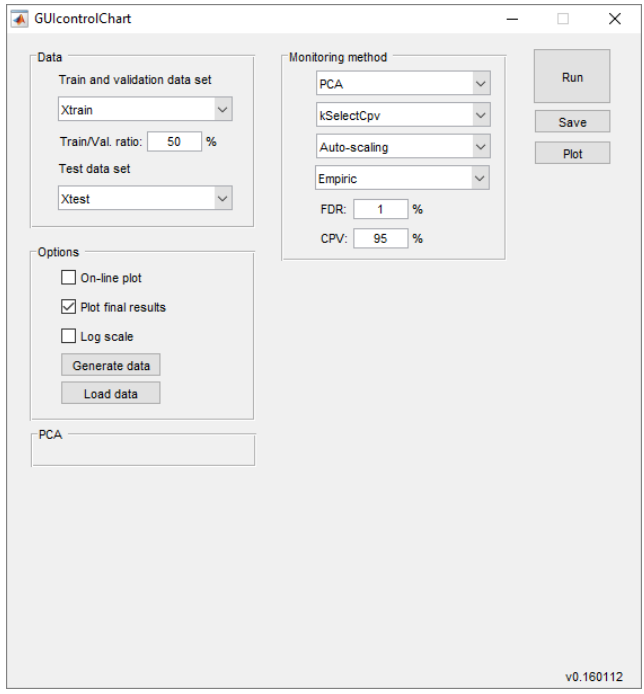

Figure 5: Screenshot of the `GUIcontrolChart` GUI after setting the modeling parameters for PCA.

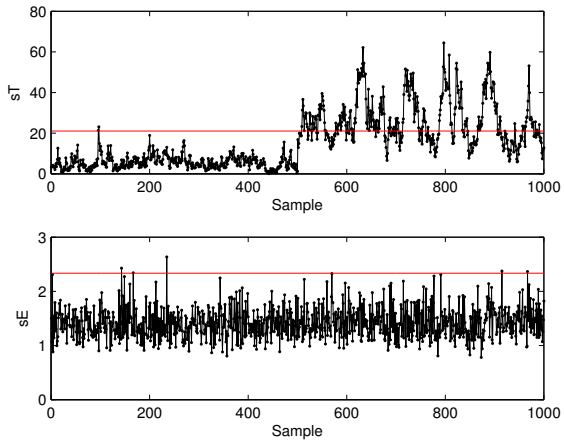

Figure 6: Plots of the PCA control charts for the AR(1) data generated using `guiTDseries`.

A.4 Real data example

Analysis will be performed on simulated data from the well-known Tennessee Eastman process [Downs and Vogel (1993)], introduced in Chapter 3, Section 3.6. Performance of methods on this simulation is a common benchmark in the SPM literature, though we choose it here as a neutral case study for our toolbox rather than with the intention to validate the methods we have implemented. The Tennsessee Eastman data is made available with the toolbox. To load this data, the "Load data" button in the **Options** section can be used. By doing so, the user is prompt to the data set location, being that the Tennsessee Eastman data is stored in "TEdata.mat". Selecting this file loads 22 data matrices into the workspace. The loaded variables are a NOC data set (`X0`) and 21 faulty data sets (`X1`, `X2`, ...,`X21`) representing the faults described in Downs and Vogel (1993). Each data set containing 960 observations collected at a sample interval of 3 minutes, with the fault introduced after 8 hours. A total of 52 variables are made available for monitoring.

A.4.1 Example: SPM using DPCA

As the data collected from the Tennsessee Eastman is characterized by dynamic features, DPCA is one of the most suitable monitoring approach for it. To perform such monitoring, `X0` is inputted as the train and validation data set.

For illustration purposes fault 7, `X7`, is selected as the test data set. A DPCA model is trained by selecting `DPCA` in the **Monitoring method** section. In this case, we opt to select the number of principal components by parallel analysis, `kSelectParAnl` and to scale the data by `Auto-scaling`. `Auto-scaling` here refers to the procedure of subtracting the process mean and dividing by the standard deviation.

Setting DPCA as the monitoring approach opens a new section in the GUI pertaining to the selection of lags. As a maximum number of lags we set a value of 10 and let the selection be made manually (i.e., using the "manual" option). Note that a lagged structures proposed by this procedure do not correspond to a single lag value to all variables, but allow different numbers of lags for the variables. The final aspect of the GUI is as presented in Figure 7. Afterwards, the control chart is run.

During the training phase of DPCA the lag selection algorithm delivers the plots of the KSV and $KSVR$ presented in Figure 8. These plots point to an optimum lagged structure at stage 416. As described in Section A.1.5, the optimality of this lagged structure is due to an almost constant KSV on the following stages and a significant drop on the $KSVR$ at that stage. By selecting this lagged structure, the control charts of Figure 9 are obtained.

The same procedure can be run using the Matlab commands below:

```
% Set modelling parameters
aG = 0.01;% False detection rate
trainValRatio = 50; % Train to validation ratio
midPoint=round(size(Xtrain,1)*trainValRatio/100); % Mid point to split
the train data set
% Modelling
model = dpcaModel( Xtrain(1:midPoint,:), 'kSelectCpv' );
% Set control limits
model = modelLimits( Xtrain(midPoint+1:end,:), model, aG );
% Monitoring
[ stat, ucl ] = dpcaMonitor( Xtest, model );
```

A.5 Glossary of functions

The Matlab toolbox **HDCC** comes with the following modelling and monitoring functions:

- `pcaModel` and `pcaMonitor`: Fit PCA model and monitor a process with it.

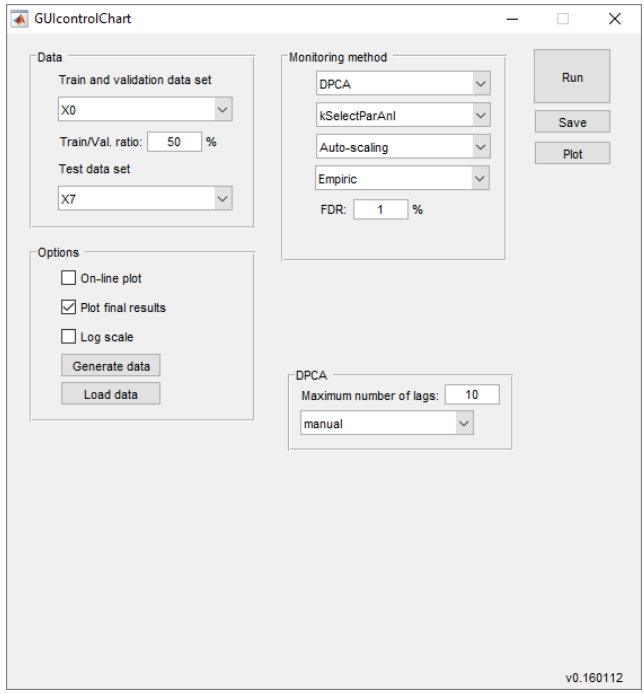

Figure 7: Screenshot of the `GUIcontrolChart` GUI after setting the modeling parameters for DPCA.

- `dpcaModel` and `dpcaMonitor`: Fit a DPCA model and monitor a process with it.
- `dpcadrModel` and `dpcadrMonitor`: Fit a DPCA-DR model and monitor a process using with it.
- `rpcaModel` and `rpcaMonitor`: Fit an RPCA model and monitor a process with it.
- `mwpcaModel` and `mwpcaMonitor`: Fit an MWPCA model and monitor a process with it.
- `mewmaModel` and `mewmaMonitor`: Fit an MEWMA model and monitor a process with it.
- `mcusumModel` and `mcusumMonitor`: Fit an MCUSUM model and monitor a process with it.

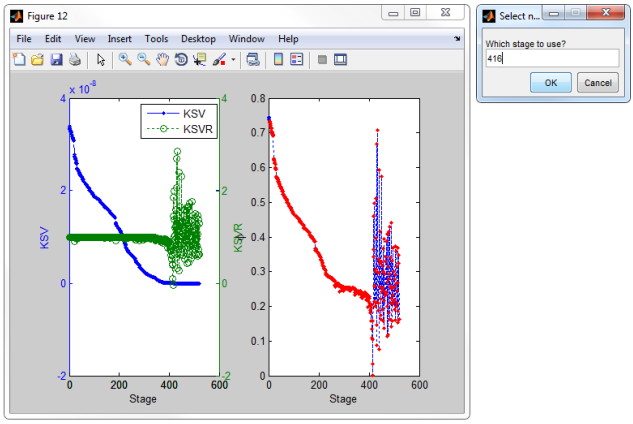

Figure 8: Screenshot of the KSV and $KSVR$ used to select the number of lags of DPCA.

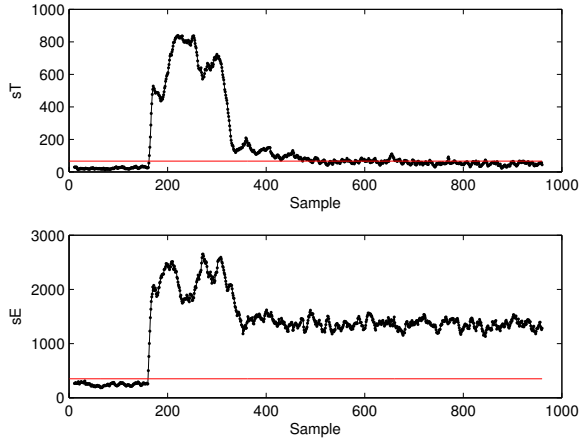

Figure 9: Plots of the DPCA control charts for fault 7 of TE data.

Functions that handle specific parameter selection and data processing tasks are:

- `kSelectCpv`, `kSelectParAnl`, and `kSelectCrossVal`: Select the number of principal components to retain using the CPV, parallel analysis and cross-validation criterion.

- `fSpeedSolve`: Identifies suitable values for the forgetting parameter in MWPCA and RPCA models.
- `lagSelect`: Selects a suitable number of lags for each variable in DPCA and DPCA-DR models.
- `modelLimits`: Set empirical control limits for PCA, DPCA and DPCA-DR models.
- `limitT` and `limitQ`: Sets theoretical control limits for the T^2 and Q-statistics for PCA-type models.
- `aSolve`: Selects α values that result in the desired FDR to use as input in the theoretical expressions for the control limits of the T^2 and Q-statistics. Used for RPCA and MWPCA models, where the α value does not always correspond exactly with the obtained FDR.
- `missingData`: Applies missing data estimation to the m variables in $\mathbf{X}$ based on a PCA model.
- `lagData`: Adds lags up to the desired order to each variable.

Functions that call model functions and produce charts and statistics as output are:

- `controlChart`: Calls modeling and monitoring functions and allows the user to apply these methods and obtain control statistics and control charts as output.
- `GUIcontrolChart`: GUI interface for using `controlChart`.
- `plotChart`: Function called by `controlChart` to create control chart plots.

Functions that generate data for simulations are:

- `TDseries`: Generates time-dependent series of type AR, MA, ARMA, IMA, ARIMA, and non-stationary in the PCA loadings to arbitrary dimensions and orders.
- `guiTDseries`: GUI interface for using `TDseries`.

Bibliography

Arteaga, F. and Ferrer, A. (2002). Dealing with missing data in MSPC: several methods, different interpretations, some examples. *Journal of Chemometrics*, 16(8-10):408–418. pages 16, 41, 144

Barceló, S., Vidal-Puig, S., and Ferrer, A. (2010). Comparison of multivariate statistical methods for dynamic systems modeling. *Quality & Reliability Engineering International*, 27(1):107–124. pages 6

Bersimis, S., Psarakis, S., and Panaretos, J. (2006). Multivariate statistical process control charts: an overview. *Quality & Reliability Engineering International*, 23:517–543. pages 5, 69

Bisgaard, S. (2012). The future of quality technology: From a manufacturing to a knowledge economy & from defects to innovations. *Quality Engineering*, 24:30–36. pages 6

Björck, Å. and Golub, G. H. (1973). Numerical methods for computing angles between linear subspaces. *Mathematics of Computation*, 27:579–594. pages 121

Box, G. (1954). Some theorems on quadratic forms applied in the study of analysis of variance problems, I. effect of inequality of variance in the one-way classification. *The Annals of Mathematical Statistics*, 25(2):290–302. pages 14, 146

Box, G., Jenkins, G. M., and Reinsel, G. (1994). *Time series analysis: forecasting and control*. Prentice-Hall, New Jersey, 3rd edition. pages 45, 47, 50, 53, 149, 150

Burnham, A. J., MacGregor, J. F., and Viveros, R. (1999). Latent variable multivariate regression modeling. *Chemometrics and Intelligent Laboratory Systems*, 48(2):167 – 180. pages 42, 71

Cadima, J. and Jolliffe, I. T. (1995). Loadings and correlations in the interpretation of principal components. *Journal of Applied Statistics*, 22:203–214. pages 111

Camacho, J., Pico, J., and Ferrer, A. (2010). Data understanding with pca: Structural and variance information plots. *Chemometrics and Intelligent Laboratory Systems*, 100(1):46–56. pages 17, 19

Camacho, J., Picó, J., and Ferrer, A. (2009). The best approaches in the on-line monitoring of batch processes based on PCA: Does the modelling structure matter? *Analytica Chimica Acta*, 642(1–2):59 – 68. pages 79, 82

Candès, E. J., Li, X., Ma, Y., and Wright, J. (2011). Robust principal component analysis? *Journal of the ACM*, 58:1–37. pages 110

Chiang, L. and Colegrove, L. (2007). Industrial implementation of on-line multivariate quality control. *Chemometrics and Intelligent Laboratory Systems*, 88:143–153. pages 34

Chiang, L., Russell, E., and Braatz, R. (2001). *Fault Detection and Diagnosis in Industrial Systems*. Springer-Verlag, London. pages 30

Choi, S., Martin, E., Morris, A., and Lee, I. (2006). Adaptive multivariate statistical process control for monitoring time-varying processes. *Industrial & Engineering Chemistry Research*, 45:3108–3118. pages 28, 31, 34, 42, 71, 78, 92, 136, 138

Crosier, R. B. (1988). Multivariate generalizations of cumulative sum quality-control schemes. *Technometrics*, 30(3):291–303. pages 142

Croux, C., Filzmoser, P., and Fritz, H. (2013). Robust sparse principal component analysis. *Technometrics*, 55:202–214. pages 110, 113, 114, 116, 117, 120, 122, 124, 127, 129, 130

Croux, C., Filzmoser, P., and Oliveira, M. R. (2007). Algorithms for projection-pursuit robust principal component analysis. *Chemometrics and Intelligent Laboratory Systems*, 87:218–225. pages 114

Croux, C. and Ruiz-Gazen, A. (2005). High breakdown estimators for principal components: the projection-pursuit approach revisited. *Journal of Multivariate Analysis*, 95:206–226. pages 110, 112

Dayal, B. S. and MacGregor, J. F. (1997a). Improved PLS algorithms. *Journal of Chemometrics*, 11(1):73–85. pages 25

Dayal, B. S. and MacGregor, J. F. (1997b). Recursive exponentially weighted PLS and its applications to adaptive control and prediction. *Journal of Process Control*, 7(3):169–179. pages 25, 34

De Ketelaere, B., Mertens, K., Mathijs, F., Sabin Diaz, D., and De Baerdemaker, J. (2011). Nonstationarity in statistical process control - issues, cases, ideas. *Applied Stochastic Models in Business and Industry*, 27:367–376. pages 101

De Ketelaere, B., Rato, T., Schmitt, E., and Hubert, M. (2016). Statistical process monitoring of time-dependent data. *Quality Engineering*, 28:127–142. pages 100

Debruyne, M. and Hubert, M. (2009). The influence function of the Stahel-Donoho covariance estimator of smallest outlyingness. *Statistics & Probability Letters*, 79:275–282. pages 139

Downs, J. and Vogel, E. (1993). A plant-wide industrial process control problem. *Computers and Chemical Engineering*, 17:245–255. pages 62, 158

Elshenawy, L. and Awad, H. (2012). Recursive fault detection and isolation approaches of time-varying processes. *Industrial & Engineering Chemistry Research*, 51(29):9812–9824. pages 136

Engelen, S. and Hubert, M. (2005). Fast model selection for robust calibration. *Analytica Chemica Acta*, 544:219–228. pages 139

Engelen, S., Hubert, M., and Vanden Branden, K. (2005). A comparison of three procedures for robust PCA in high dimensions. *Austrian Journal of Statistics*, 34:117–126. pages 115

Epprecht, E. K., Barbosa, L. F. M., and Simões, B. F. T. (2011). SPC of multiple stream processes: A chart for enhanced detection of shifts in one stream. *Production*, 21:242 – 253. pages 8

Epprecht, E. K. and Simões, B. F. T. (2013). Statistical control of multiple-steam processes–a literature review. *Paper presented at the 11th International Workshop on Intelligent Statistical Quality Control, Sydney, Australia.* pages 8

Ferrer, A. (2007). Multivariate statistical process control based on principal component analysis (MSPC-PCA): Some reflections and a case study in an autobody assembly process. *Quality Engineering*, 19(4):311–325. pages 37, 69

Ferrer, A. (2014). Latent structures-based multivariate statistical process control: A paradigm shift. *Quality Engineering*, 26(1):72–91. pages 69

Filzmoser, P., Fritz, H., and Kalcher, K. (2014). *pcaPP: Robust PCA by Projection Pursuit*. version 1.9-50. pages 114

Folch-Fortuny, A., Villaverde, A., Ferrer, A., and Banga, J. (2015). Enabling network inference methods to handle missing data and outliers. *BMC Bioinformatics*, 16(1):283. pages 16

Gallagher, N., Wise, B., Butler, S., White, D., and Barna, G. (1997). Development and benchmarking of multivariate statistical process control tools for a semiconductor etch process: Improving robustness through model updating. In *Process: Impact of Measurement Selection and Data Treatment on Sensitivity, Safe Process 1997*, pages 26–27. pages 25

He, B. and Yang, X. (2011). A model updating approach of multivariate statistical process monitoring. In *2011 IEEE International Conference on Information and Automation (ICIA)*, pages 400–405. pages 31, 32

He, X. and Yang, Y. (2008). Variable MWPCA for adaptive process monitoring. *Industrial & Engineering Chemistry Research*, 47(2):419–427. pages 26, 31, 138

Horn, J. (1965). A rationale and test for the number of factors in factor analysis. *Psychometrika*, 30(2):179–185. pages 143

Hubert, M. and Engelen, S. (2007). Fast cross-validation for high-breakdown resampling algorithms for PCA. *Computational Statistics & Data Analysis*, 51:5013–5024. pages 139

Hubert, M., Rousseeuw, P., and Vanden Branden, K. (2005). ROBPCA: a new approach to robust principal components analysis. *Technometrics*, 47:64–79. pages 15, 34, 101, 110, 112, 113, 114, 121, 128

Hubert, M., Rousseeuw, P. J., and Verboven, S. (2002). A fast method for robust principal components with applications to chemometrics. *Chemometrics and Intelligent Laboratory Systems*, 60:101–111. pages 110

Hubert, M., Rousseeuw, P. J., and Verdonck, T. (2009). Robust PCA for skewed data and its outlier map. *Computational Statistics & Data Analysis*, 53(6):2264 – 2274. pages 139

Hubert, M. and Vanden Branden, K. (2003). Robust methods for partial least squares regression. *Journal of Chemometrics*, 17:537–549. pages 139

Jackson, J. and Mudholkar, G. (1979). Control procedures for residuals associated with principal component analysis. *Technometrics*, 21(3):341–349. pages 13, 14

Jeng, J. C. (2010). Adaptive process monitoring using efficient recursive PCA and moving window PCA algorithms. *Journal of the Taiwan Institute of Chemical Engineer*, 44:475–481. pages 30, 31

Jensen, W., Birch, J., and Woodall, W. (2007). High breakdown estimation methods for phase I multivariate control charts. *Quality and Reliability Engineering International*, 23(5):615–629. pages 34

Jin, H., Lee, Y., Lee, G., and Han, C. (2006). Robust recursive principal component analysis modeling for adaptive monitoring. *Industrial & Engineering Chemistry Research*, 45(20):696–703. pages 31

Jolliffe, I. (2002). *Principal Component Analysis*. Springer, New York, 2nd edition. pages 11, 143

Jolliffe, I. T., Trendafilov, N. T., and Uddin, M. (2003). A modified principal component technique based on the LASSO. *Journal of Computational and Graphical Statistics*, 12:531–547. pages 17, 110, 112, 127, 129

Jose, C. (2014). Visualizing big data with compressed score plots: Approach and research challenges. *Chemometrics and Intelligent Laboratory Systems*, 135:110–125. pages 34

Kourti, T. (2005). Application of latent variable methods to process control and multivariate statistical process control in industry. *International Journal of Adaptive Control and Signal Processing*, 19(4):213–246. pages 6, 15, 37, 87

Kruger, U. and Xie, L. (2012). *Advances in Statistical Monitoring of Complex Multivariate Processes: with Applications in Industrial Process Control*. John Wiley, New York. pages 6, 28, 30, 37, 136

Kruger, U., Zhou, Y., and Irwin, G. (2004). Improved principal component monitoring of large-scale processes. *Journal of Process Control*, 14(8):879–888. pages 21, 74

Krzanowski, W. J. and Kline, P. (1995). Cross-validation for choosing the number of important components in principal component analysis. *Multivariate Behavioral Research*, 30(2):149–165. pages 144

Ku, W., Storer, R., and Georgakis, C. (1995). Disturbance detection and isolation by dynamic principal component analysis. *Chemometrics and Intelligent Laboratory Systems*, 30(1):179–196. pages 20, 21, 22, 42, 67, 71, 73, 93, 144, 148

Lakshminarayanan, S., Shah, S., and Nandakumar, K. (1997). Modeling and control of multivariable processes: dynamic PLS approach. *American Institute of Chemical Engineers Journal*, 43(9):2307–2322. pages 42, 71, 93, 148

Lee, J., Qiu, H., Yu, G., Lin, J., and Services, R. T. (2007). Bearing data set. *IMS, University of Cincinnati. NASA Ames Prognostics Data Repository*. pages 7

Lemberge, P., De Raedt, I., Janssens, K. H., Wei, F., and Van Espen, P. J. (2000). Quantitative z-analysis of the 16–17th century archaelogical glass vessels

using PLS regression of EPXMA and μ-XRF data. *Journal of Chemometrics*, 14:751–763. pages 128

Li, G. and Chen, Z. (1985). Projection-pursuit approach to robust dispersion matrices and principal components: Primary theory and Monte Carlo. *Journal of the American Statistical Association*, 80:759–766. pages 110

Li, W., Yue, H., Valle-Cervantes, S., and Qin, S. (2000). Recursive PCA for adaptive process monitoring. *Journal of Process Control*, 10(5):471–486. pages 25, 34

Locantore, N., Marron, J. S., Simpson, D. G., Tripoli, N., Zhang, J. T., and Cohen, K. L. (1999). Robust principal component analysis for functional data. *Test*, 8:1–73. pages 110

Lowry, C., Woodall, W., Champ, C., and Rigdon, S. (1992). A multivariate exponentially weighted moving average control chart. *Technometrics*, 34(1):46–53. pages 142

Luo, R., Misra, M., and Himmelblau, D. (1999). Sensor fault detection via multiscale analysis and dynamic PCA. *Industrial & Engineering Chemistry Research*, 38(4):1489–1495. pages 22

Miller, P., Swanson, R., and C., H. (1998). Contribution plots: A missing link in multivariate quality control. *Applied Mathematics and Computer Science*, 8:775–792. pages 18

Montgomery, D. (2008). *Introduction to Statistical Quality Control*. Wiley Desktop Editions Series. Wiley. pages 73, 89

Nelson, P. R., Taylor, P. A., and MacGregor, J. F. (1996). Missing data methods in PCA and PLS: Score calculations with incomplete observations. *Chemometrics & Intelligent Laboratory Systems*, 35(1):45 – 65. pages 41, 144

Nomikos, P. and MacGregor, J. (1995). Multivariate SPC charts for monitoring batch processes. *Technometrics*, 37:41–59. pages 13

Oppenheim, A. V. and Schafer, R. W. (1975). *Digital signal processing*. Prentice-Hall, New York. pages 84

Pfeffermann, D. and Allon, J. (1989). Multivariate exponential smoothing: Method and practice. *International Journal of Forecasting*, 5(1):83 – 98. pages 89

Qin, J., Valle-Cervantes, S., and Piovoso, M. (2001). On unifying multi-block analysis with applications to decentralized process monitoring. *Journal of Chemometrics*, 15:715–742. pages 19

Qin, S. (2003). Statistical process monitoring: Basics and beyond. *Journal of Chemometrics*, 17:480–502. pages 6, 19, 136

R Core Team (2014). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. pages 16, 117

Ramaker, H., van Sprang, E., Westerhuis, J., and Smilde, A. (2005). Fault detection properties of global, local and time evolving models for batch process monitoring. *Journal of Process Control*, 15(7):799 – 805. pages 79, 88

Rato, T. and Reis, M. (2013a). Advantage of using decorrelated residuals in dynamic principal component analysis for monitoring large-scale systems. *Industrial & Engineering Chemistry Research*, 52(38):13685–13698. pages 22

Rato, T. and Reis, M. (2013b). Defining the structure of DPCA models and its impact on process monitoring and prediction activities. *Chemometrics and Intelligent Laboratory Systems*, 125:74–86. pages 22, 43, 72, 73, 145, 155

Rato, T. and Reis, M. (2013c). Fault detection in the Tennessee Eastman benchmark process using dynamic principal components analysis based on decorrelated residuals (DPCA-DR). *Chemometrics and Intelligent Laboratory Systems*, 125:101–108. pages 22, 41, 144

Rato, T., Schmitt, E., Reis, M., De Ketelaere, B., and Hubert, M. (2016). A systematic comparison of PCA-based statistical process monitoring methods for high-dimensional, time-dependent processes. *AIChE Journal*, 62(5):1478–1493. pages 79, 83

Rousseeuw, P. J. (1984). Least median of squares regression. *Journal of the American Statistical Association*, 79:871–880. pages 113

Rousseeuw, P. J. and Croux, C. (1993). Alternatives to the median absolute deviation. *Journal of the American Statistical Association*, 88:1273–1283. pages 113

Runger, G. C., Alt, F. B., and Montgomery, D. C. (1996). Controlling multiple stream processes with principal components. *International Journal of Production Research*, 34(11):2991–2999. pages 8

Russell, E., Chiang, L., and Braatz, R. (2000). Fault detection in industrial processes using canonical variate analysis and dynamic principal component analysis. *Chemometrics and Intelligent Laboratory Systems*, 51:81–93. pages 37, 47, 62, 64, 88

Schmitt, E., Rato, T., Reis, M., De Ketelaere, B., and Hubert, M. (2016). Parameter selection guidelines for adaptive pca-based control charts. *Journal of Chemometrics*, 30(4):163–176. pages 73

Serneels, S. and Verdonck, T. (2008). Principal component analysis for data containing outliers and missing elements. *Computational Statistics & Data Analysis*, 52(3):1712 – 1727. pages 16, 34

Todorov, V. (2013). *Scalable Robust Estimators with High Breakdown Point for Incomplete Data*. R package, version 0.4-4. pages 16

Todorov, V. (2014). *rrcovHD: Robust Multivariate Methods for High Dimensional Data*. version 0.2-3. pages 114, 121

Todorov, V. and Filzmoser, P. (2009). An object-oriented framework for robust multivariate analysis. *Journal of Statistical Software*, 32:1–47. pages 16, 121

Todorov, V. and Filzmoser, P. (2013). Comparing classical and robust sparse PCA. In Kruse, R., Berthold, M. R., Moewes, C., Gil, M. Á., Grzegorzewski, P., and Hryniewicz, O., editors, *Synergies of Soft Computing and Statistics for Intelligent Data Analysis*, number 190 in Advances in Intelligent Systems and Computing, pages 283–291. Springer-Verlag, Berlin Heidelberg. pages 114, 121

Treasure, R. J., Kruger, U., and Cooper, J. E. (2004). Dynamic multivariate statistical process control using subspace identification. *Journal of Process Control*, 14(3):279 – 292. pages 24

Tsung, F. (2000). Statistical monitoring and diagnosis of automatic controlled processes using dynamic PCA. *International Journal of Production Research*, 38(3):625–637. pages 21

Valle, S., Li, W., and Qin, S. (1999). Selection of the number of principal components: The variance of the reconstruction error criterion with a comparison to other methods. *Industrial & Engineering Chemistry Research*, 38(11):4389–4401. pages 11, 143

van Sprang, E. N., Ramaker, H.-J., Westerhuis, J. A., Gurden, S. P., and Smilde, A. K. (2002). Critical evaluation of approaches for on-line batch process monitoring. *Chemical Engineering Science*, 57(18):3979 – 3991. pages 79

Vanden Branden, K. and Hubert, M. (2005). Robust classification in high dimensions based on the SIMCA method. *Chemometrics and Intelligent Laboratory Systems*, 79:10–21. pages 139

Vanhatalo, E. and Kulachi, M. (2015). The effect of autocorrelation on the Hotelling T^2 control chart. *Quality and Reliability Engineering International, 10.1002/qre.1717*. pages 74, 81

Verboven, S. and Hubert, M. (2005). LIBRA: a Matlab Library for Robust Analysis. *Chemometrics and Intelligent Laboratory Systems*, 75:127–136. pages 16

Walczak, B. and Massart, D. (2001). Dealing with missing data. part I. *Chemometrics & Intelligent Laboratory Systems*, 58:15—27. pages 16, 34

Wang, X., Kruger, U., and Irwin, G. (2005). Process monitoring approach using fast moving window PCA. *Industrial & Engineering Chemistry Research*, 44(15):5691–5702. pages 32

Wikstrom, C., Albano, C., Eriksson, L., Johansson, E., Sandberg, M., Kettaneh-Wold, N., Fridén, H., Rännar, S., Nordahl, A., and Wold, S. (1998). Multivariate process and quality monitoring applied to an electrolysis process: Part II. multivariate time-series analysis of lagged latent variables. *Chemometrics and Intelligent Laboratory Systems*, 42:233–240. pages 70, 75, 76

Wold, S. (1994). Exponentially weighted moving principal components analysis and projections to latent structures. *Chemometrics and Intelligent Laboratory Systems*, 23(1):149–161. pages 25, 90

Woodall, W. and Montgomery, D. (2014). Some current directions in the theory and application of statistical process monitoring. *Journal of Quality Technology*, 46(1):78–94. pages 6, 69, 136

Xie, L., Kruger, U., Lieftucht, D., Littler, T., Chen, Q., and Wang, S.-Q. (2006). Statistical monitoring of dynamic multivariate processes part 1. modeling autocorrelation and cross-correlation. *Industrial & Engineering Chemistry Research*, 45(5):1659–1676. pages 41

Zhou, Z., Li, X., Wright, J., Candès, E. J., and Ma, Y. (2010). Stable principal component pursuit. *Proceedings of International Symposium on Information Theory*. pages 110

Zou, H., Hastie, T., and Tibshirani, R. (2006). Sparse principal component analysis. *Journal of Computational and Graphical Statistics*, 15:265–286. pages 17, 110

List of publications

Publications

Schmitt, E., Rato, T., Reis, M., De Ketelaere, B., Hubert, M. (2016). Parameter selection guidelines for adaptive PCA-based control charts. *Journal of Chemometrics*, 30(4), 163–176. DOI: 10.1002/cem.2783

De Ketelaere, B., Rato, T., Schmitt, E., Hubert, M. (2016). Statistical process monitoring of time-dependent data. *Quality Engineering*, 28(1), 127–142. DOI: 10.1080/08982112.2015.1100474

De Ketelaere, B., Schmitt, E., Rato, T., Hubert, M. (2016). Statistical process monitoring of time-dependent data: Rejoinder. *Quality Engineering*, 28(1), 151–154.

Schmitt, E., Vakili, K. (2016). The FastHCS algorithm for robust PCA. *Statistics and Computing*. 26(6), 1229–1242. DOI: 10.1007/s11222-015-9602-5

Hubert, M., Reynkens, T., Schmitt, E., Verdonck, T. (2015). Sparse PCA for high-dimensional data with outliers. *Technometrics. Accepted.* DOI: 10.1080/00401706.2015.1093962

Rato, T., Schmitt, E., De Ketelaere, B., Hubert, M. Reis, M. (2015). A Systematic Comparison of Statistical Process Monitoring Methods for High-dimensional, Time-dependent Processes. *AIChE Journal*, 62(5), 1478–1493. DOI:10.1002/aic.15062

Hubert, M., De Ketelaere, B., Schmitt, E. (2015). Overview of PCA-based statistical process monitoring methods for time-dependent, high-dimensional data. *Journal of Quality Technology*, 47(4), 318–335.

Schmitt, E. Öllerer, V., Vakili, K. (2014). The finite sample breakdown point of PCS. *Statistics & Probability Letters*, 94, 214–220. DOI: 10.1016/j.spl.2014.07.026

Vakili, K., Schmitt, E. (2014). Finding multivariate outliers with FastPCS. *Computational Statistics & Data Analysis*, 69(1), 54–66. DOI: 10.1016/j.csda.2013.07.021

Marasigan, V., Schmitt, E., Garmyn, M., Van den Noortgate, W. (2014). A meta-analysis on major risk factors of multiple primary cutaneous melanomas. *Clinical Dermatology*, 2(2), 84–92. DOI: 10.11138/cderm/2014.2.2.084

Vandecandelaere, M., Schmitt, E., Vanlaar, G., De Fraine, B., Van Damme, J. (2014). Effects of kindergarten retention for at-risk children's mathematics development. *Research Papers in Education*. DOI: 10.1080/02671522.2014.919523

Vandecandelaere, M., Schmitt, E., Vanlaar, G., De Fraine, B., Van Damme, J. (2014). Effects of kindergarten retention for at-risk children's psychosocial development. *Educational Psychology: An International Journal of Experimental Educational Psychology*. DOI: 10.1080/01443410.2014.950194

Presentations

Tull, C., Schmitt, E., Atwater, P. How Much Water Does Turf Removal Save? Applying Bayesian Structural Time-Series to California Residential Water Demand. SIGKDD. San Francisco, USA, 13-17, September 2016.

Schmitt, E., Vakili, K. Robust PCA with FastHCS. ENBIS. Prague, Czech Republic, 6-10, September 2015.

Schmitt, E., De Ketelaere, B., Rato, T., Reis, M. The Challenges of PCA-Based Statistical Process Monitoring: An Overview and Solutions. ENBIS. Linz, Austria, 21-25 September 2014.

Schmitt, E. Leong, W.C. But is it Probably Art? Reasons (not) to define art. An international conference on the relevance of defining art. Gent, Belgium. 19-20 May 2014.

Schmitt, E., Rato, T., Reis, M., De Ketelaere, B., Hubert, M. A Systematic Comparison of Statistical Process Monitoring Methods for High-dimensional, Time-dependent Processes. ERCIM. London, UK, 14-16 December 2013.

Schmitt, E., Rato, T., Reis, M., De Ketelaere, B. A Systematic Comparison of Statistical Process Monitoring Methods for High-dimensional, Time-dependent Processes. ENBIS. Ankara, Turkey, 14-19 September 2013.

External report

Vandecandelaere, M., Schmitt, E., Vanlaar, G., De Fraine, B., Van Damme, J. (2014). Zittenblijven in de derde kleuterklas: Effecten op de psychosociale ontwikkeling van kinderen. Leuven: Steunpunt Studie- en Schoolloopbanen.

Vandecandelaere, M., Vanlaar, G., De Fraine, B., Van Damme, J., Schmitt, E. (2013). Kindergarten retention in Flanders: differential effects on mathematics growth for gender and language groups. Leuven: Steunpunt SSL.

Working Papers

Atwater, P., Schmitt, E., Atwater, D. (2015). Towards California water conservation impact evaluations by default: lessons of a turf removal rebate study in South Orange County. *In review.*

Reluga, K. and Schmitt, E. (2015). Recapture of water investments in nutrients cycle using *Hermetia illucens.*

Vakili, K. and Schmitt E. (2013). Finding Regression Outliers with FastRCS. arXiv:1307.4834 [stat.ME].

Rato, T. and Schmitt, E. (2016). HDCC: The Matlab Toolbox for Multivariate and High-dimensional Control Charts.

FACULTY OF SCIENCE
DEPARTMENT OF MATHEMATICS
ROBUST @ LEUVEN
Celestijnenlaan 200B
B-3001 Leuven
https://wis.kuleuven.be/stat/robust